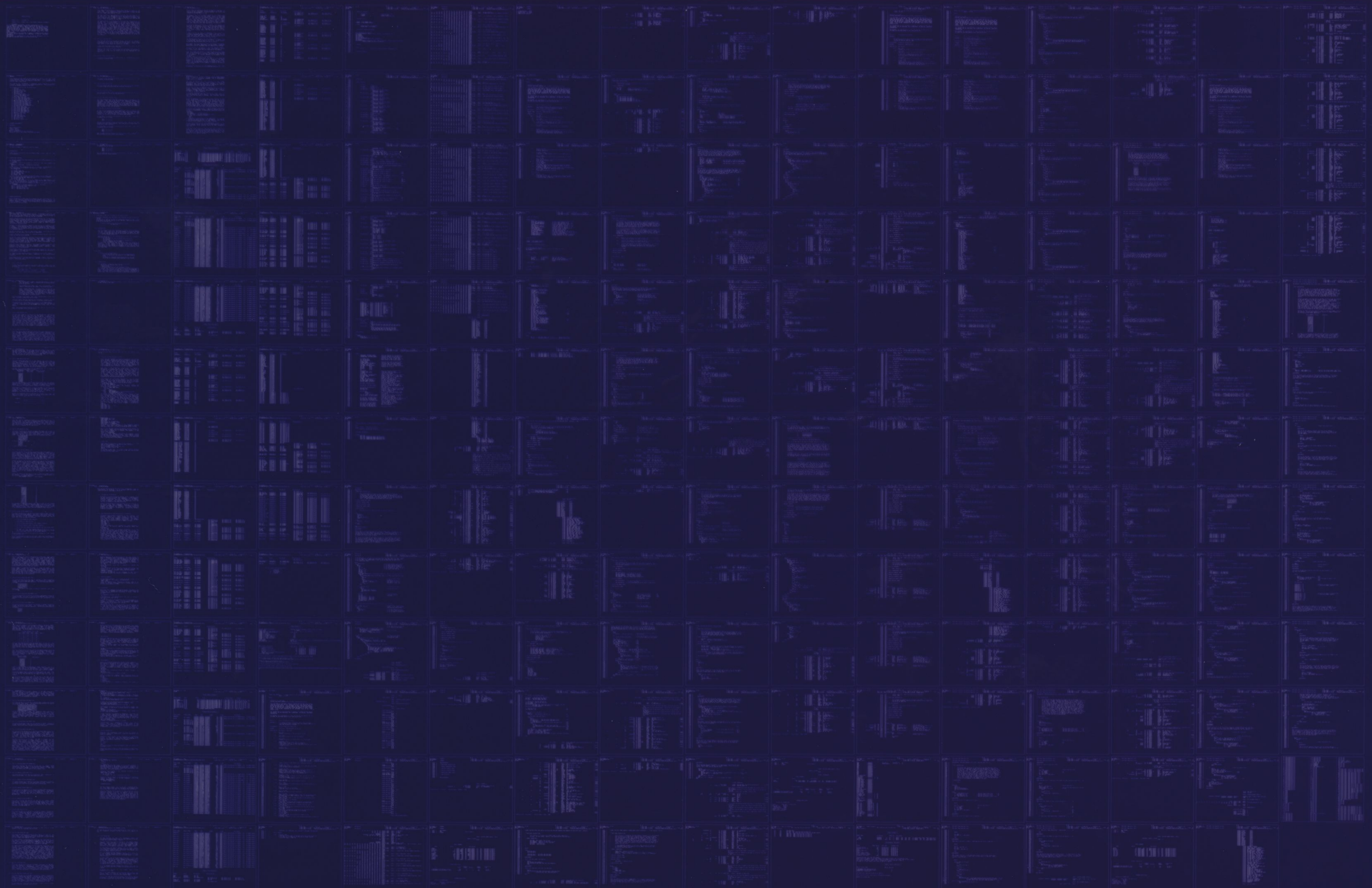
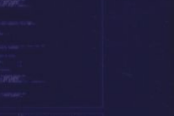
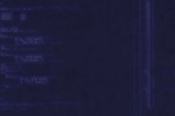
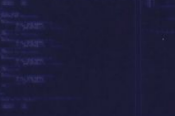
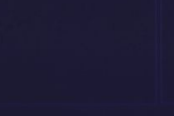
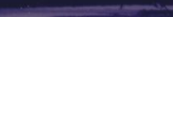
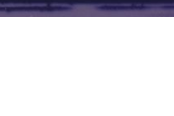
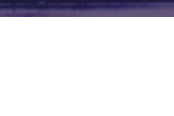
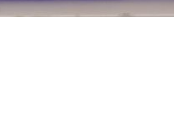
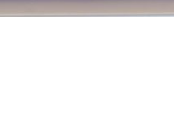
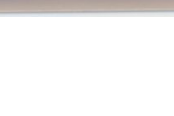




IDENTIFICATION

PRODUCT ID: ZZ-ECCBA-1.4

PRODUCT TITLE: ECCBA13 VAX 11/750 UNIBUS INTERFACE DIAGNOSTIC

DECO/DEPO: 1.4

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
REMAIN IN DEC.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

This program provides detection and isolation of faults in the VAX 11/750 Unibus Interface logic of the VAX 11/750 UBI module. This is a macro level program which runs under the Diagnostic Supervisor. It is intended for use only on a VAX 11/750 system.

The program was implemented in BLISS-32, which requires a slightly different approach in using the program documentation. This document will provide the needed guidance to the user.

The program consists of the following tests:

1. CSR Test
2. Map Data Bus Test
3. Map Chip Select Test
4. Map Address Bus Test
5. Map Entry Test
6. CPU Read/Write Test
7. CMI to Unibus Address Test
8. Unibus to CMI Address Test
9. Data Path Select Test
10. Direct Data Path DATI Test
11. Direct Data Path DATIP/DATO Test
12. Direct Data Path DATOB Test
13. Buffered Address Register Test
14. Buffered Data Path DATI Test
15. Buffered Data Path DATIP Test
16. Buffered Data Path DATO Test
17. Buffered Data Path DATOB Test
18. Buffered Data Path Autopurge Test
19. Byte Offset DATI Test
20. Byte Offset DATIP/DATO Test
21. Byte Offset DDP DATO Test
22. Byte Offset BDP DATO Test
23. Byte Offset DDP DATOB Test
24. Byte Offset BDP DATOB Test
25. Map Entry Functional Test
26. CSR Status Bit Test
27. Interrupt Test
28. CSR2 Interrupt Test
29. CSR1/CSR2 ACLOW Test
30. Map Invalid Test
31. UET PB Bit Test
32. UBE Block Transfer Test

2.0 HARDWARE REQUIREMENTS

2.1 EQUIPMENT

VAX-11/750 Processor
256kb of memory
Console terminal
Console load device
One or two DW750's with UET terminators.
Optionally one to four Unibus Exercisers per DW750.

The VAX-11 Diagnostic Supervisor must be loaded.

4.0 PREREQUISITES

EVKAA Hardcore Instruction Test and EVKAB & EVKAC Cluster Exerciser must run successfully.

5.0 LOADING AND STARTING

To load the program from TU58 cassette, type the following commands on the VAX 11/750 console:

```
>>>B DDA0
```

The Supervisor takes a few seconds to initialize, and then outputs the prompt

```
DIAGNOSTIC SUPERVISOR. ZZ-ECSAA-V6.X-XXX DATE
DS>
```

on the console. The program may be run in various configurations. To start the program in its default configuration, simply type

```
DS> @ECCBA <CR>
```

or

```
DS> LO[AD] ECCBA <CR>
```

```
DS> ATT[ACH] DW750 CMI DWn <CR>
```

```
DS> SET T[RACE] <CR>
```

```
DS> SEL[ECT] DWn <CR>
```

```
DS> ST[ART] <CR>
```

This will run all tests except for the last test. To run that test, you must first attach the Unibus Exerciser(s) with the following command:

```
DS> ATT[ACH] UBE DW0 UBn 7700x0 5xx <CR>
```

```
DS> SEL[ECT] UBn <CR>
```

Where n is the UBE number (0,1,2, or 3), 7700x0 is the CSRO address, and 5xx is the interrupt vector. If you have more than one UBE on the system, each one must be attached and selected for test.

A normal, error-free run will produce a printout on the console similar to the following:

```
.. PROGRAM: VAX 11/750 UBI Diagnostic, REV 1.0, 30 TESTS.
```

```
TEST 1: Control and Status Register Test
```

```
TEST 2: Map Data Bus Test
```

```
TEST 3: Map Chip Select Test
```

```
TEST 4: Map Address Bus Test
```

```
.  
.
.  
.
.
```

6.0 PROGRAM DESCRIPTION

Before attempting to understand this program, the user should have a firm grasp of the functional characteristics and hardware logic of the CUI. An adequate amount of background should be provided by the hardware manual and circuit schematics. However, a brief description of some characteristics of the CUI will be given here.

Three main functions are provided by the CUI. It provides a way for the VAX 11/750 CPU to access Unibus registers, it provides a way for Unibus devices to perform DMA transfers with VAX 11/750 memory, and it provides a way for Unibus devices to interrupt the VAX 11/750 CPU.

Several characteristics of the VAX architecture and the VAX 11/750 memory subsystem have influenced the design of the CUI. First of all, contiguous virtual addresses may be physically discontinuous (on 512 byte boundaries). However, Unibus NPR devices always create sequential addresses during transfers. Therefore, the CUI must be able to break up these transfers into distinct 512 byte blocks.

Secondly, the VAX architecture imposes no restrictions on alignment of data in memory. Unibus NPR devices only transfer word data on even addresses. To accomodate this difference, the CUI provides a method for the transfer to be effectively shifted by one byte for transfers to I/O buffers aligned on an odd byte address.

The CUI also has to be able to map Unibus space, which has 18 bit addresses, into CMI space, which has 24 bit addresses.

Finally, the CMI acts as a buffer for transfers between the two byte wide Unibus and the four byte wide CMI.

The CUI has a set of diagnostic registers which are used in all tests that require a transfer between CMI and Unibus. These diagnostic registers are similar in functionality to the Unibus Exerciser (UBE). However, this 'mini-UBE', which resides on the Time-of-Year clock module, is able only to perform single word NPRs (the UBE is able to perform programmable length transfers from 1 word to 64k words).

There are three diagnostic registers: the diagnostic address register, the diagnostic data register, and the diagnostic control register. These should be fully described in the CUI maintenance manual. Basically, they function in the following manner.

The diagnostic address register contains the lower 16 bits of the 18 bit Unibus address. The upper two bits of the Unibus address are contained in the diagnostic control register.

The diagnostic data register contains the data to write to CMI memory, or the data received from CMI memory, depending on the type of Unibus transaction.

The diagnostic control register contains bits which control the type of Unibus transaction, the interrupt level (BR4-BR7), a GO bit, and bits 16 and 17 of the Unibus address.

6.1 Test 1: CSR Test

In this test the three control and status registers (CSRs) for the buffered data paths (BDPs) are tested. These registers are organized as follows:

	31	30	29	28		1	0
	+-----+-----+-----+-----+-----+-----+-----+-----+						
	!Error! NXM !		UCE !		not used		!Purge!
	+-----+-----+-----+-----+-----+-----+-----+-----+						
Bit 31	Error - Logical OR of bits 29 and 30. Read only.						

- Bit 30 Non existent memory (NXM) - Set when NXM status is received from CMI memory. SSYN is withheld from the Unibus device. Write one to clear.
- Bit 29 Uncorrectable error (UCE) - Set when uncorrectable error status is received for CMI memory. Write one to clear.
- Bit 0 Purge - Always reads zero. Writing a zero to it has no effect. Writing a one to it produces a result based on the contents of the buffer. With unibus data in the buffer, the data is written to the CMI and the byte flags are set to mark the buffer empty. With CMI data in the buffer, the byte flags are set to mark the buffer empty. If the buffer is already empty, no action occurs.

Bits 28 through 1 are unused, and yield unpredictable values when read. Writing to these bits has no effect.

This test checks that the registers are accessible, readable, and that writing a one to the status bits results in the bits being cleared.

The registers are each probed with a routine called PROBE_ADDRESS. If a machine check occurs, the program reports the error, but does not abort. It is possible to put the program into a scope loop if this happens.

Next, the program checks that bits 0, 29, 30, and 31 are all zero after writing ones to bits 29 and thirty.

6.2 Test 2: Map Data Bus Test

This test the integrity of a portion of the map data bus (that portion of the data path to and from the map which is used for CMI read/writes). The map is made up of 256 X 4 RAM ICs, organized into two rows of 5 ICs. In this test, a pattern of all ones is written to a map in the first row of RAMs, and then to a map in the second row. Any bit that is clear in both rows is determined to be a stuck at zero fault (although it might also be a compound fault with that RAM bit stuck at zero and the chip select line also stuck, for instance). Then, a pattern of all zeros is written to a RAM in each of the two rows. Any bit that is set in both rows is determined to be a stuck at one fault. If each of the nineteen bits can toggle from 1 to 0, the map data bus visible to the program is considered to be good.

6.3 Test 3: Map Chip Select Test

This test checks that the map chip select line is not stuck at one or zero. The map registers are made up of 256 X 4 RAMs, organized in two rows of 5 RAMs. The map chip select line is really a row select line, with the row select bit corresponding to the value of bit 10 of the map CMI address. To test whether this line is stuck, we write a unique value to a map entry with bit 10 at zero, and then we write a unique value to a map entry with bit 10 at one. If the chip select line is stuck at one or zero, the second value will overwrite the first. In order to guard against RAM data bit failures causing misleading results in this test, we write all ones to map entry 0, and

all zeros to map entry 256, and then count the number of ones in each map upon reading them. If the number of ones in a map is less than 9, the entry is considered to represent a zero. If the number of ones in the map entry is 9 or more, the entry is considered to represent a one. In this way, the test tolerates up to 9 RAM data bit failures before the results of the test are misrepresented.

6.4 Test 4: Map Address Bus Test

This test checks that there are no map address bus bits stuck or shorted. A unique pattern is written into each map entry at the map entry addresses 0, 1, 2, 4, 8, 16, 23, 64, and 128 (so that each address bit assumes both a zero and a one). Reading each of these map entries after all patterns are written will reveal any stuck or shorted address bus bits, as the last pattern written to a doubly addressed map will overwrite the earlier pattern written.

Consider the following example with map address bus bit 3 stuck at zero (the addresses are given in binary):

Map address bits 7:0	Data stored bits 14:0	Data retrieved bits 14:0
00000000	0	4
00000001	1	1
00000010	2	2
00000100	3	3
00001000	4	4
00010000	5	5
00100000	6	6
01000000	7	7
10000000	8	8

Note that the contents of entry 0 were overwritten with the pattern stored at entry 8 (1000 base 2), since these two entries coincide when address bit 3 is stuck at zero. In general, an address bit n stuck at 0 will show up as $n+1$ being stored in entry 0 with this scheme.

Unfortunately, a data bit fault in one of the RAMs in a particular map entry may point erroneously to an address bus bit fault. Therefore, to make this test insensitive to RAM data bit failures, it is necessary to encode each stored value prior to storing. The Reed-Muller (16,5) error-correcting code is used here.

Because the codewords that are used here are sixteen bits long, and there is no single field in a map entry having 16 contiguous bits, the codeword is stored in the map entry as follows:

	15	1	0
codeword	! ! ! !	! !	!
	21 14		0
map entry	! ! ! !	!	!

This tests that all readable/writeable bits of each map entry are not stuck at zero or one. This is accomplished by writing all zeros to all map entries, then reading each map entry in turn. After verifying that the entry contains all zeros, all ones are written to the entry, and then the entry is read again to verify that it now contains all ones. This is repeated for all map entries.

6.6 Test 6: CPU Read/Write Test

This tests the ability of the CUI to handle CPU reads and writes to the Unibus. The diagnostic data and address registers are written and read, checking for stuck at one or zero faults. In this way, the diagnostic registers are also tested before they are used in later tests. The following data patterns are used to perform this test:

```
0101010101010101
1010101010101010
0011001100110011
0000111100001111
0000000011111111
```

to check for word writes, and

```
01010101
10101010
00110011
00001111
```

to check for byte writes.

6.7 Test 7: CMI To Unibus Address Test

This tests the ability of the CUI to correctly map a CMI address to a Unibus address in a Unibus read (DATI) transaction. Three types of fault could occur here: the first, a stuck or shorted bit in the map entry or data paths leading to the map; the second, failure to assert the PFN field of the map entry as the CMI page address; and the third, a stuck or shorted bit in the byte number address path. The first type of fault was dealt with in the map entry test. Therefore, this test deals with the second and third varieties of faults.

Subtest one deals with the case of the CUI failing to put the contents of the map PFN onto the CMI as a memory page address. The diagnostic data register is cleared with a CPU write, and a unique data pattern is written to a CMI location. The CUI and diagnostic registers are set up to access that location, and a DATI is initiated. After a nominal wait, the diagnostic data register is read. If the register contains the correct data, then this subtest is successful. However, we do an additional DATI at this time with Unibus address bit 1 changed, in order to determine if that Unibus address bit is stuck. Since the high word of the data was different from the low word, this should result in the contents of the diagnostic data register changing.

Subtest two deals with the case of a fault in the byte number field of the address. Unique data patterns are stored in a page of memory at the following byte number addresses:

BYTE ADDRESS (binary)	DATA (DECIMAL)
-----------------------	----------------

-----	----
000000000	0
000000100	2
000001000	3
000010000	4
000100000	5
001000000	6
010000000	7
100000000	8
111111100	10
111111000	12
111110100	13
111101100	14
111011100	15
110111100	16
101111100	17
011111100	18

This sequence of addresses causes a one and a zero to be floated through bits 2 through 8 of the byte number field (bits 0 and 1 are treated with separate logic). A DATI transaction is initiated to read each location in turn. Any stuck or shorted address bit will result in the wrong data being written into the diagnostic data register.

6.8 Test 8: Unibus To CMI Address Test

This tests the ability of the CUI to correctly select a map from the page frame of the Unibus address. The type of fault which could occur here is a stuck or shorted Unibus address bit, which would cause the wrong map entry to be selected. In order to detect this type of fault, the following steps are executed:

1. Clear the diagnostic data register.
2. Invalidate all maps but one.
3. Write a unique data pattern into a memory location.
4. Set up and execute a DATI from that location.
5. Verify that the diagnostic data register has the correct data.
6. Do steps 1 to 5 for every useable map of the set 0, 1, 2, 4, 16, 32, 64, 128, and 256 (this causes a one to be floated across each page frame bit of the Unibus address).

The byte number field need not be explicitly tested, as it was tested in the previous test.

The direct data path is used for this test. Note that the presence of Unibus memory will preclude testing some subset of the Unibus address page frame bits.

This test detects any stuck or shorted bits in the data path select lines. The PURGE bit is written in each of the three BDP CSRs to ensure that the buffers are empty. Three DATO cycles are then initiated, with three unique data patterns, selecting each of the three different BDPs. With the three buffers loaded, another DATO transaction is initiated with a fourth unique pattern, selecting the DDP. Each BDP is purged in turn, and each written memory location is verified to contain the proper data pattern. Any stuck or shorted bit in the data path select lines will be clearly shown by one of the data patterns occurring in more than one location.

Before the test is executed, the Unibus is sized for memory. Any Unibus memory present will cause map entries accessed with the same Unibus page frame to be unuseable. Therefore, before any testing requiring may functionality commences, the Unibus is sized, and a bit map of useable maps is built by the Unibus sizer routine. Any test requiring the use of one or more maps selects the maps to use on the basis of th entries in this bitvector.

6.10 Test 10: Direct Data Path DATI Test

This tests the ability of the CUI to perform a DATI transaction through the direct data path (DDP). The transactions are made form longword and word aligned CMI addresses. The following data patterns are used:

```
0101010101010101
1010101010101010
0011001100110011
0000111100001111
0000000011111111
```

Before testing starts, the Unibus is sized to determine which maps are useable.

6.11 Test 11: Direct Data Path DATIP/DATO Test

This tests the ability of the CUI to perform a DATIP/DATO transaction through the DDP. This test is similar to the DDP DATI test, with the addition of checking that the DATO portion of the transaction also works.

6.12 Test 12: Direct Data Path DATOB Test

This tests the ability of the CUI to perform a DATOB transaction through the direct data path. The transactions are made from longword and word aligned CMI addresses. The following data patterns are used:

```
01010101
10101010
00110011
00001111
```

6.13 Test 13: Buffered Address Register Test

This tests the integrity of the buffered address registers (BARs). These three registers, one per BDP, are located in the UDP chips. They are bit sliced, with four bits per UDP. For diagnostic reference, the UDPs are assigned the logical designations UDP0, UDP1, UDP2, and UDP3. The Unibus address bits corresponding to each chip are as follows:

Unibus address bits in UDP					
UDP#	BAR3	BAR2	BAR1	BAR0	BAR bit
0	16	15	03	02	
1	08	17	10	09	
2	05	04	12	11	
3	07	06	14	13	

There are two types of fault that can occur here. In the first type the BAR shows an address match when there should not be one, and in the second type of error the BAR fails to indicate a match when there should be one. The first two subtests deal with the first variety of fault, and the second two subtests deal with the second variety of fault.

The first subtest deals with the case of the address match occurring when there should not be a match, due to a stuck or shorted bit in the byte number field (bits 2 through 8) of the BAR. In order to detect this condition, a sequence of unique data patterns is written into a page of memory at the following sequence of byte number addresses:

CMI byte number address (binary)	data pattern
000000000	0
000000100	2
000001000	3
000010000	4
000100000	5
001000000	6
010000000	7
100000000	8

A DATI is then executed to read each of these patterns, one at a time, through each of the three BDPs in turn. The contents of the diagnostic data register are then verified. This address sequence causes a one to be floated across the byte number field of the BAR.

The second subtest causes a one to be floated across the page frame field of the BAR by executing DATIs with each map of the set 0, 1, 2, 4, 8, 16, 32, 64, 128, and 256 (for all maps of that set that are useable).

The third and fourth subtests deal with the case of the BAR failing to indicate an address match when it should. In order to detect this type of fault, it is necessary to execute a DATI from a particular CMI location, change the data in that CMI location, and execute the DATI again without changing the address. If the BAR properly indicates an address match, the data in the diagnostic register would

not change after the first DATI. However, if the BAR contained a stuck or shorted bit which caused a failure to indicate a match in this case, the data in the diagnostic data register would change after the second DATI, revealing the fault. In order to check each bit of the BAR, this procedure is followed for the same addressing patterns as in the preceding two subtests.

6.14 Test 14: Buffered Data Path DATI Test

This test the ability of the CUI to perform DATI transactions through the buffered data paths. The following data patterns are used to verify the integrity of the data paths:

```
01010101010101010101010101010101
10101010101010101010101010101010
00110011001100110011001100110011
00001111000011110000111100001111
00000000111111110000000011111111
00000000000000001111111111111111
```

The following algorithm is used. The diagnostic data register is cleared, then a DATI is executed with address bit 1 at a zero. After a nominal wait, the data in the diagnostic data register is verified. Then a second DATI is executed with address bit 1 at a one, and the data in the diagnostic data register is verified again. This is repeated for each BDP, and for longword and word aligned CMI references.

6.15 Test 15: Buffered Data Path DATIP Test

This test the ability of the CUI to perform a DATIP transaction through the three BDPs. There should be no difference in performance between a DATIP and a DATI through the buffered data paths.

6.16 Test 16: Buffered Data Path DATO Test

This test verifies the ability of the CUI to handle DATO transactions through each of the buffered data paths (BDPs). The CUI behavior for a DATO is primarily dependent on the contents of the buffer. The first case is when the buffer has Unibus data, and no address match in the BAR. The second case is when the buffer is empty or contains CMI data. The third case is when the buffer has Unibus data, and there is an address match in the BAR.

The test algorithm is as follows. The BDP is purged, two CMI locations are cleared, a unique pattern is written into the diagnostic data register, and a DATO is executed. The data is then changed in the diagnostic data register, and a second DATO is set up and executed with a different destination address (outside the initial longword). After the first DATO, check that the first CMI destination is still at its initialized value. After the second DATO, check that the first CMI destination contains the the first unique pattern. Execute a third DATO, changing Unibus address bit 1, and also changing the data in the diagnostic data register. Check that the second CMI location now contains the second and third patterns, concatenated into one longword. This effectively tests the

6.17 Test 17: Buffered Data Path DATOB Test

This test verifies the ability of the CUI to handle DATOB transactions through each of the buffered data paths (BDPs). This test is essentially identical to the one just before it (BDP DATO Test), but checks that only byte writes to memory occur.

6.18 Test 18: Address Offset DATI Test

This tests the ability of the CUI to perform a DATI transaction in address offset mode. All four data paths are used.

6.19 Test 19: Address Offset DATI/DATO Test

This tests the ability of the CUI to perform a DATIP/DATO transaction in address offset mode. All four data paths are used. However, the DATO occurs only when the direct data path is being used.

6.20 Test 20: Address Offset DATO Test

This test verifies the ability of the CUI to handle DATO transactions through each of the BDPs with byte offset mode enabled. The behavior of the CUI in this mode is similar to its behavior in a normal DATO, with the exception that odd byte addressing of the CMI is accomplished. The algorithm for this test is similar to that of the BDP DATO Test.

6.21 Test 21: Address Offset DATOB Test

This test verifies the ability of the CUI to handle DATOB transactions through each of the buffered data paths with byte offset mode enabled. This test is similar to the BDP DATOB test, except that in this test the CUI's ability to address odd CMI locations is verified. The test algorithm here is similar to the algorithm employed in the BDP DATOB Test.

6.22 Test 22: Map Function Test

This tests the ability of the CUI to correctly handle transactions using all useable map registers. 496 map registers are potentially useable, but of that set some may not be useable due to the presence of Unibus memory. Therefore, before the testing of the maps is started, the Unibus is sized for memory to ascertain which of the maps are useable. A bitvector is generated by the Unibus sizer routine. If a map is useable, its bit in the bitvector is set.

For each map that is available for testing, a DATI with a unique pattern is set up, executed, and verified.

6.23 Test 23: CSR Status Bit Test

This tests that there are no stuck at zero faults in each of the three CSRs. The diagnostic features of the memory controller are used to simulate the uncorrectable error status bit. To test the NXM bit, an illegal nexus address is used.

The first subtest is concerned with verifying the functionality of the NXM bit. The CUI and diagnostic registers are set up to execute a DATI from a non-existent CMI nexus (F40000 for now). The DATI is initiated, and after a nominal wait, the CSR is read, checking that both NXM and Error are set. This is repeated for each BDP.

The second subtest is concerned with verifying the functionality of the UCE bit. A data pattern is written into memory with ECC disabled. This causes the check bits for this pattern to be written into the memory controller CSR1 <06:00>. ECC is enabled, and a second pattern is written into the same location, simulating a double-bit error. The CUI and diagnostic registers are set up for a DATI from this memory location. The memory controller is then put into diagnostic check mode and page mode. This causes the check bits to be drawn from the memory controller CSR 1 <06:00> instead of memory, so that an uncorrectable error will be indicated when the location in question is read. A DATI is initiated, which should cause an uncorrectable error to be indicated by the memory controller. After a nominal wait, the CUI CSR is read, checking that both UCE and Error are set. This is repeated for each of the three BDPs.

6.24 Test 24: Interrupt Test

This tests the ability of the CUI to handle interrupts at BR4-BR7, using the diagnostic registers to initiate interrupts at each BR level. The CUI has a programmable vector (the diagnostic data register), which for the first subtest is arbitrarily chosen as 340 (hex). Subtest 2 floats a one and a zero thru the interrupt vector and verifies that the vector is transmitted to the CPU correctly.

6.25 Test 25: Map Entry Functional Test

This tests the ability of the UBI to correctly handle transactions using all useable map registers. 496 map registers are potentially useable, but of that set some may not be useable due to the presence of Unibus memory. Therefore, before the testing of the maps is started, the Unibus is sized for Unibus memory to ascertain which of the maps are useable. A bitvector is generated by the Unibus sizer routine. If a map is useable, its bit in the bitvector is set.

For each map that is available for testing, a DATI with a unique pattern is set up, executed, and verified.

This tests that there are no stuck at zero faults in each of the three CSRs. The diagnostic features of the memory controller are used to simulate the uncorrectable error status bit. To test the NXM bit, an illegal nexus address is used.

The first subtest is concerned with verifying the functionality of the NXM bit. The UBI and UET are set up to execute a DATI from a non-existent CMI nexus (F40000 for now). The DATI is initiated, and after a nominal wait, the CSR is read, checking that both NXM and ERR are set. This is repeated for each BDP.

The second subtest is concerned with verifying the functionality of the UCE bit. A data pattern is written into memory with ECC disabled. This causes the check bits for this pattern to be written into the memory controller CSR 1 <06:00>. ECC is enabled, and a second pattern is written to the same location, simulating a double-bit error. The UBI and UET are set up for a DATI from this memory location. Then the memory controller is put into Diagnostic Check Mode and Page Mode. This will cause the check bits to be drawn from the memory controller CSR 1 <06:00> instead of memory, so that an uncorrectable error will be indicated when the location in question is read. A DATI is initiated, which should cause an uncorrectable error to be indicated by the memory controller. After a nominal wait, the UBI CSR is read, checking that both UCE and ERR are set. This is repeated for each BDP. The PE bit in the UET is also checked.

6.27 Test 27: Interrupt Test

This tests the ability of the UBI to handle interrupts at BR4-BR7, using the UET to initiate interrupts at each BR level. The UET has a programmable vector, which for this test is arbitrarily chosen as 340 (hex), which is just below the vectors for the UBEs.

If the interrupt occurs, the interrupt service routine clears the BR request bit in the UET Control Register.

Subtest 5 then floats a one and a zero thru the vector and forces an interrupt on level 4. All vectors for the Unibus are captured so that failures in the interrupt vector will not be reported by the Supervisor.

6.28 Test 28: CSR2 Interrupt Test

This test is only executed for DW750 number 1. The DW750 must not be in external processor mode.

6.29 Test 29: CSR1/CSR2 ACLOW Test

This test is only executed for DW750 number 1 and only if it is not in external processor mode.

6.30 Test 30: Map Invalid Test

This test checks that UBI refuses to issue SSYN if the UET attempts to access a map entry which has the valid bit turned off.

6.31 Test 31: UET PB Bit Test

This test checks that if PB is ascertained on the Unibus, the CPU will machine check if it tries to read the Unibus.

6.32 Test 32: UBE Block Transfer Test

This test supports up to four UBEs, each of which is assigned a CUI data path. Each UBE is set up to perform large block transfers at high speed (10 microseconds per word). All transfers are started together, such that a maximum of bus activity takes place. The program selects the appropriate routine to use based on the number of UBEs selected. As large a transfer size as possible (up to 128kb) is used for each transfer (the available free buffer space and number of UBE's selected is the limiting factor).

For one UBE, 8 transfers are performed.

For the first 4 transfers, the 4 data paths are used in rotation, one per map, in the sequence DDP, BDP1, BDP2, BDP3, with byte offset mode disabled. For the second 4 transfers, a single data path is used exclusively for each transfer, with byte offset mode enabled. Each transfer (of the set of 4) uses a different data pattern.

For two UBEs, 16 transfers are executed for each UBE. Data Path assignments are as follows:

UBE0: 0, 1, 2, 3
UBE1: 1, 2, 3, 0

UBE0 performs transfers without byte offset mode enabled and UBE1 performs with byte offset enabled. Each combination of Data Paths is tested with 4 data patterns.

For three UBEs, 16 transfers are executed for each UBE. Data Path assignments are as follows:

UBE0: 0, 1, 2, 3
UBE1: 1, 2, 3, 0
UBE2: 2, 3, 0, 1

UBE0 and UBE2 perform transfers with byte offset mode disabled and UBE1 performs with byte offset mode enabled.

For four UBEs, 16 transfers are performed with each UBE. Data path assignments are as follows:

UBE0: 0, 1, 2, 3
UBE1: 1, 2, 3, 0
UBE2: 2, 3, 0, 1
UBE3: 3, 0, 1, 2

UBE1 and UBE3 operate with byte offset disabled and UBE0 and UBE2 operate with byte offset enabled.

1.2 Fixed bugs in UBE test related to one memory array.

1.3 The words "EXPECTED" and "RECEIVED" were reversed on some of the error reports.
The INIT bit in CSR1 of the UET will sometimes read as a one. This would cause the UET_ISR routine to INIT the UET. Modified ISR routine to not write the INIT bit.

A Guide to Reading the CUI Diagnostic Listing
by Rand Gray

8.0 INTRODUCTION

The CUI diagnostic has been implemented in BLISS, and this means to the user that there will be a few changes in using the program listing. The changes should not prove difficult to adapt to, and it is hoped that the user, once he has become familiar with BLISS, will find the changes beneficial.

Among the goals set for this implementation of the CUI diagnostic were the following:

1. The user need not have any BLISS training to use the program.
2. The user should find the diagnostic documentation entirely adequate for his needs.

To those ends, the machine code listing was made as readable as possible. In most cases, the user can entirely ignore the BLISS source code listing. However, if the user is willing to invest a small amount of time reading this document, he will find that using the BLISS source code listing will enhance his understanding of the test algorithms.

9.0 GOALS

The goals of this document are:

1. To help the user find his way to the needed machine code listing without the necessity for understanding BLISS.
2. To provide an adequate BLISS background for the user who wishes an enhanced understanding of the program.

10.0 ORGANIZATION

This document is organized into three main sections:

How to Use the Machine Code Listing

A BLISS Primer

A Detailed Explanation of the Program Listing

The first section will assist the user with no BLISS background who has neither the time nor inclination to acquire any knowledge of BLISS. The second section provides the user with a foundation in BLISS principles which will enable him to benefit from the BLISS source code. The last section provides a detailed explanation of

ZZ-ECCBA-1.4

Documentation

F 2

Fiche 1 Frame F2

Sequence 18

each section of the diagnostic program listing, from the first page
of the listing to the last.

This section describes how to find and use the machine code listing.

11.1 Locating The Machine Code Listing

Entry into the diagnostic listing is provided by the console error message. Unlike some previous diagnostic programs, the VAX-11 series of diagnostics do not call out a failing PC. The reason for this is that the diagnostic implementor does not know what the PC will be (this is established at link time). However, in the error printout the test, subtest, and error number are all called out. This will enable the user to find the machine code listing of the failing subtest.

Each test in the diagnostic listing has a functional description, followed by the BLISS source code, which is followed by the machine code listing. The first step is to locate the test in the listing. The tests appear in sequential order in the listing: Test 1, Test 2, Test 3, and so on. Locate and read the functional description of the failing test. This is a very important step, and should not be skipped. Quite often taking the time to read the functional description of a test can give the user a better perspective of the encountered problem.

After reading the functional description, scan the BLISS source code for the failing subtest. Once this is found, continue looking through the BLISS source code until the error number is found. This is quite obvious, as the error number will appear in the listing in the following format:

```
; %PRINT ERROR NUMBER N
```

Go to the line immediately following the error number, and note the source line number. For example, consider the following listing fragment:

```
; 1511 IF .TEMP1 NEQ .TEMP2
; 1512 THEN
; 1513 BEGIN
; %PRINT: ERROR NUMBER 1
; 1514 $DS_ERRHARD(UNIT=.LUN,MSG_ADR=RCS_CPU_RW);
; 1515 $DS_PRINTX(FMT_CPU_RW);
```

The source line number of interest is 1514. Advance through the listing until you locate the machine code listing for this test. The machine code listing can be found immediately following the BLISS source code listing. Scan the extreme right of the listing until you locate the source line number (1514 in our example). All of the machine code associated with that line appears sequentially in the listing, starting at the line identified with the source line number, and continuing until the next source line number is encountered. Here is a machine code listing fragment illustrating our example:

```
00561 BEQL 2$ ;
00563 CLRQ -(SP) ; 1514
00565 CLRQ -(SP) ;
00567 CLRQ -(SP) ;
00569 CLRL -(SP) ;
```

Documentation

```
0056B PUSHAB RCS_CPU_RW ;
0056F MOVZBL LUN, -(SP) ;
00574 PUSHL 1 ;
00576 CALLS 10, @ DS$ERRHARD ;
0057D PUSHAB FMT_CPU_RW ; 1515
00581 CALLS 1, @ DS$PRINTX ;
```

The column of five-digit numbers is the assembler location counter. Note that from location counter 00563 to 00576 is code associated with line 1514 of the BLISS source listing.

Now, to find the beginning of the subtest, having located the error call, simply scan the machine code listing in reverse until the first BGNSUB call is located. This appears in the machine code listing as

```
CALLS 2, @ DS$BGNSUB
```

There are block comments describing the program action interspersed with the machine code. These comments, coupled with the functional description of the test, should provide the user with an adequate amount of information to understand what is taking place in the program.

11.2 Scope Loops

Scope loops are available in most of the subtests. A scope loop typically appears in the machine code listing as:

```
BBS #7, @ DSA$GL_FLAGS+1, N$
for a relatively short scope loop, and
BBC #7, @ DSA$GL_FLAGS+1, M$
BRW N$
```

for a longer scope loop. In either case, search backward through the machine code listing for the N\$ label and you have the limits of the scope loop.

This section should provide the user with a fundamental level of understanding of BLISS-32. It is assumed that the reader has little or no prior knowledge of BLISS-32.

12.1 Introduction To BLISS-32

BLISS-32 is a block-structured, medium level language which has many features of a high level language. It uses algebraic notation for calculations, with operations for arithmetic, shifting, comparison, and logic. It supports a rich variety of data structures, and provides the developer with the facility to create his own data structures.

However, BLISS-32 does not have any built-in I/O routines as do typical high level languages. BLISS-32 is a machine dependent language, and this permits a high degree of coupling between the software and hardware. It is this feature of BLISS-32 which makes it suitable for implementing hardware diagnostic programs.

12.2 Data Representation And Manipulation In BLISS-32

The default data element in BLISS-32 is the longword. The BLISS-32 compiler will always attempt to use the most efficient machine code to handle a calculation, and this is often done by using longword instructions. For example, consider the following BLISS-32 expressions (the variable names represent consecutive bytes of memory storage):

```
BYTE_ONE = 0;  
BYTE_TWO = 0;  
BYTE_THREE = 0;  
BYTE_FOUR = 0;
```

The BLISS-32 compiler, in all probability, would generate a single line of machine code to accomplish those four assignments:

```
CLRL BYTE_ONE
```

Of course, many times in a hardware diagnostic it is not desirable to allow the compiler free choice. For instance, if we wish to perform a read or write to or from a Unibus device, we must be sure that the compiler will not try to use a longword reference (which would cause a trap). This is easily accomplished in BLISS-32. Any size of bus reference can be caused to happen simply by an explicit specification of the field size of the operand. In general, the compiler will use branch-on-bit instructions in dealing with single-bit fields, byte instructions for byte aligned 8 bit fields, and word instructions for (at least) byte aligned 16 bit fields.

12.3 The Fetch Operator

Because addresses can be manipulated in the same manner as data in BLISS-32, there is an operator which distinguishes a value in an expression as either an address or the contents of the address. This operator is represented as a period. In BLISS-32, a variable name by itself is always an address. A period preceding a variable name represents the contents of that address. For example, the expression

$X = 123;$
stores the decimal value 123 in location X, whereas the expression

$.X = 123;$
stores the decimal value 123 in the location specified by the contents of X. As an additional example, consider the representation of an increment instruction in BLISS-32:

$COUNTER = .COUNTER + 1;$
This expression adds one to the contents of the location COUNTER and replaces the contents of COUNTER with the sum.

12.4 Value Assignments

The assignment operator " $=$ " is used to assign a value to a storage location. For example, the expression

$TEMP = 1000;$
causes the longword value representing decimal 1000 to be stored in the four consecutive bytes of storage beginning at the byte whose address is TEMP.

12.5 Expressions

Most high level languages distinguish between statements and expressions. A statement performs an action without producing a value, and an expression calculates a value. Thus an assignment such as

$A = 1;$
is a statement, whereas the sum

$.B + 1;$
is an expression. The combination of the two is typically permitted on the same line as

$A = .B + 1;$
in which the value of the expression $.B + 1$ is assigned to A. However, in BLISS-32, there is no distinction made between statements and expressions. Thus, an assignment can legally appear anywhere in an expression, such as

$X = .Y * (A = .B + 1);$
computes the value $.B + 1$ and multiplies it by the contents of Y. The resulting value is stored in location X. However, at the same time, with no additional computation, the value $.B + 1$ was stored in location A. When it is desired to terminate an expression for which a value is not to be associated, the expression is terminated in a semicolon.

Blocks provide for the organization of a program into units. The typical block is delimited by the pair of keywords BEGIN and END. Blocks can also be legally delimited by a pair of parentheses. A block delimited by a pair of parentheses is generally termed a parenthesized expression. A block which contains no declarations is generally termed a compound expression. There are two main categories of blocks: blocks that compute a value, and blocks that only take action.

12.7 Declarations

Before any name may be used in a BLISS-32 program, it must be declared. It is the initial declaration of a name that provides the compiler with information about the attributes to be associated with the name. A declaration is altogether distinct from an expression. For instance, the declaration

```
LITERAL A = 1234;
```

means that whenever A appears in the program, the value 1234 decimal is to be used. The value 1234 is NOT stored in location A.

Storage is allocated to a program with a declaration such as

```
OWN BETA;
```

In this example, four consecutive bytes are allocated to the program, and the name BETA is associated with the address of the first byte. A declaration may also specify attributes to be associated with a name, as in the following example:

```
OWN STATUS: VECTOR[4];
```

which allocates four longwords of storage and tells the compiler that STATUS has the attribute of a longword vector.

12.8 Data Segments

Values in a BLISS-32 program are stored in data segments. A data segment is defined as one or more bytes of memory. There are two main categories of data segments: scalars and structures. A scalar is a single value, and can be either a byte, word, or longword in length. It may also be signed or unsigned. The default attributes are longword, unsigned. For instance,

```
OWN ALPHA;
```

allocates four consecutive bytes of memory storage. In contrast,

```
OWN ALPHA: BYTE;
```

allocates a single byte of memory storage.

There are several very commonly used structures which are defined as part of BLISS-32. These predeclared structures include:

1. Vectors
2. Bitvectors
3. Blocks
4. Blockvectors

A vector structure is a sequence of elements. The number of

Documentation

elements is the extent of the vector, and is given as part of the declaration of the data segment name. Thus,

```
OWN ALPHA: VECTOR[3];
```

allocates three consecutive longwords of storage. Reference to a particular element in the program would typically appear as

```
B = .ALPHA[1];
```

This expression would store the contents of the second element into the location whose address is B.

A vector can also be declared with an allocation unit. This allocation unit can be byte, word, or longword. Thus,

```
OWN GAMMA: VECTOR[11,BYTE];
```

allocates eleven consecutive bytes of storage.

The bitvector structure is similar to the vector, except that each element of the vector is always one bit in length. For example,

```
OWN DELTA: BITVECTOR[15];
```

allocates two consecutive bytes of storage, even though one bit will never be referenced.

A block structure is a sequence of components. The block has a name, and each component of the block has a name. A block is declared with a size and an allocation unit. The size will specify the total amount of required storage for the entire block. The allocation unit specifies the units in which the size is measured. Thus,

```
OWN IOTA: BLOCK[6,BYTE];
```

allocates six consecutive bytes of storage.

The blockvector is a sequence of elements (just like a vector). The number of elements is the extent of the blockvector, and is given as part of the declaration of the segment name. Each element of the blockvector is a sequence of components (just like a block).

12.9 Control Expressions

There are five fundamental kinds of flow of control in BLISS-32: sequential, conditional, iterative, subroutine, and condition handling.

Sequential flow of control is the evaluation of expressions strictly in the order that the expressions appear in the program. This applies to expressions that are contained in a block and separated by semicolons. For example:

```
BEGIN
```

```
A = .X + 1;
```

```
B = .A*3;
```

```
C = 24;
```

```
END;
```

This block consists of three assignments, each of which are evaluated in turn.

Conditional flow of control is the evaluation of either one or another expression on the basis of the outcome of a test. For example:


```
IF .EXPECTED EQL .RECEIVED  
THEN  
    FLAG = 1  
ELSE
```

```
    FLAG = 0;  
This expression sets FLAG equal to 1 if the contents of  
EXPECTED are equal to the contents of RECEIVED. Otherwise, it  
sets FLAG equal to 0. BLISS-32 has three main categories of  
conditional flow expressions: conditional expressions, case  
expressions, and select expressions.
```

Iterative flow of control is the repeated evaluation of an expression. For example:

```
INCR I FROM 0 TO 99 DO  
    A[I] = B[I];
```

In this example, the loop is evaluated 100 times.

Subroutine flow of control is the evaluation of an expression (usually a block) that is written somewhere else in the program. For example:

```
ROUTINE INCREMENT(ALPHA) =  
    ALPHA = .ALPHA + 1;
```

```
    . . .  
    INCREMENT(ALPHA);  
    . . .
```

When the program reaches the routine-call INCREMENT(ALPHA), the flow jumps to the routine, evaluates the assignment, and returns the flow to the next line following the routine call.

Condition handling flow of control is the evaluation of an expression (usually a block) in response to an unusual condition detected either by the VAX-11 hardware or by software. The expression that is evaluated, which must be coded as the body of a routine, is determined by the stack of routine calls that lead to the detection of the condition and different routines can be invoked for the same condition at different times. Flow of control may or may not return to the point at which the condition was detected.

13.0 A DETAILED EXPLANATION OF THE PROGRAM LISTING

This section provides an in-depth explanation of each section of the CUI diagnostic program listing, from the first page to the last.

13.1 The Program Map

The map provides the user with information about the attributes of program modules and provides addresses of global storage and values of global literals.

The first thing to be found in the map is the OBJECT MODULE SYNOPSIS. This provides a list of all of the modules that were linked together to form the executable program image. There are four modules involved: CUI_HEADER, CUI_ISR, CUI_TESTS, and a Diagnostic Supervisor symbol table, DS.STB.

For each module, the version number, size in bytes, object file name (including disk, user identification code, and file version number), creation date, and creator (the program that built the object module) are given.

Next, the PROGRAM SECTION SYNOPSIS is given. This lists all of the program sections (P-SECTs) that appear in all of the modules. The same P-SECT name can (and does) appear in more than one module.

For each P-SECT, the following things are given. First, the module name identifies which module this particular P-SECT resides in. Note that the P-SECTs \$CODE\$ and \$TESTCNT\$ are to be found in two modules. These are, in fact, not related at all (except in name). Next, the base address of the P-SECT, its ending address, and its length in bytes is given. This is followed by its alignment, and finally its attributes.

It is the PROGRAM SECTION SYNOPSIS that enables the user to locate the actual position in memory of a given portion of the program.

Following this section is the symbol table sorted alphabetically by name.

Next is the IMAGE SYNOPSIS, which gives the total size of the program image, its starting and ending address, and some additional statistics concerning the program image.

Finally, the LINK RUN STATISTICS appear, which chiefly provide performance information on various items including the amount of CPU and elapsed time required to link the program image.

13.2 The CUI_HEADER Module

This is the first module of the CUI diagnostic program. It chiefly contains the common subroutines and utilities that are required by the diagnostic. It also contains the error and information message ASCII strings.

The first line of this module declares the module name (CUI_HEADER) and its version number, given in the format 'NN-MM'. Every page of this module will have a two-line page header. On the first line of this header will be the module name, its creation date and time, the file creator (VAX-11 Bliss-32), and the page number of the listing. The second line of the page header gives the program version number, creation date and time, and the source file name (including disk, user identification code, and source file version).

Following the module declaration is a BEGIN, which denotes the beginning of the module. Following this are the standard DEC copyright notice and disclaimers. Next, the module's facility is given, followed by a terse abstract of the module's functionality. Finally, the program's environment, author, source file creation date, and modification history are provided.

On page 2 resides the TABLE OF CONTENTS, which provides a listing of all of the routines to be found in the module. The declaration FORWARD ROUTINE simply means that any routine listed can call any other routine in the list even if it appears in the module after the calling routine.

The INCLUDE FILES section lists the library and require files that are needed for compilation of the module. A library file is precompiled, whereas a require file is in source format, and must be compiled along with the source module.

Next appear the five diagnostic supervisor macros, \$DS_BGNMOD, \$DS_BITDEF, \$DS_DSDEF, \$DS_DSADEF, and \$DS_HPODEF. These are macros which are all expanded from the require file DIAG.B32.

The next section of the module is entitled EQUATED SYMBOLS. Following is the declaration LITERAL, which declares all of the literal symbol names that are required by this module.

After the LITERAL declaration is found the declaration GLOBAL BIND, which declares all of the symbol names to be associated with the ASCII strings for messages, and also for UBE register bit assignments. A typical message string appears in the listing as

```
EXP_MSG = UPLIT BYTE(ASCIC ' EXPECTED'),
```

A PLIT is a Pointer to a Literal. A UPLIT is an uncounted PLIT. The attribute BYTE tells the compiler to allocate the declared string in bytes rather than in longwords. The ASCIC keyword is a directive to form a counted ASCII string. Finally, the message appears between the quotation marks. The symbolic name (EXP_MSG in this example) is bound to the address of the string.

Following this section is the OWN STORAGE section. The first declaration in this section is a STRUCTURE declaration. This

Documentation

declaration creates a data structure called `ERROR_BUFFER`, which is used to define a storage area for data compare errors. This is structured somewhat like a blockvector, except that it is simpler.

The next declaration in the listing is the `GLOBAL` declaration. This creates storage allocation, much as the declaration `OWN`, except that the storage is globally accessible. Note that the declaration of `UBE_ERR_BUF` uses the `ERROR_BUFFER` structure that was just declared.

Next in the listing is a `MACRO` declaration, which is used to declare several names as macros. These macros are similar to assembly language macros in that whenever the compiler encounters a name previously declared as a macro, it treats the name as its macro expansion.

The `EXTERNAL REFERENCES` section includes any names that are declared outside of this module. In this case, there are four. They are the names of the four `UBE` interrupt service routines.

On the next page of the listing the first routine of this module is declared. It is declared a global routine, so that it will appear in the program map. The routine declaration includes the name of the routine and any parameters which are used by the routine. In the case of this routine, `PRINT_CSR_ERR`, there are three formal parameters: `CSR_NUM`, `EXP_DATA`, and `REC_DATA`. Following the routine declaration is a routine header, which contains the functional description of the routine and other pertinent information about the routine.

The routine itself begins with the `BEGIN` statement immediately following the routine header. The first expression of the routine is a `CODECOMMENT`. A `CODECOMMENT` expression always has the form

`CODECOMMENT`

'The comment is in between quotes.'

'And additional lines are separated by commas.'

'The last line is ended by a colon.'

`BEGIN`

`END;`

The code that is associated with the `CODECOMMENT` is inserted in between the `BEGIN` and `END` of the expression. The compiler is then constrained to put all of the machine code inline following a block comment in the assembly listing.

The code associated with the first `CODECOMMENT` consists of two keyword macros (which are defined in the diagnostic require file, `DIAG.B32`) which generate calls to the Diagnostic Supervisor.

Following the `BLISS` source listing of the first routine is found the first piece of the machine code listing. This includes all of the own storage that was generated, as well as the machine code listing for the first routine. Also appearing in this section of the machine code listing are any equates (in this case all global).

+-----+
! Object Module Synopsis !
+-----+

Module Name	Ident	Bytes	File	Creation Date	Creator
UBI_HEADER	01-04	6012	[LEMIEUX.ECCBA.RELEASE]ECCBA0.OBJ;28	6-Sep-1985 17:17	VAX-11 Bliss-32 V4.1-746
UBI_SUBROUTINES	01-04	1612	[LEMIEUX.ECCBA.RELEASE]ECCBA1.OBJ;11	6-Sep-1985 17:18	VAX-11 Bliss-32 V4.1-746
UBI_ISR	01-04	410	[LEMIEUX.ECCBA.RELEASE]ECCBA2.OBJ;12	6-SEP-1985 17:18	VAX/VMS Macro V04-00
UBI_TESTS_1_5	01-04	1691	[LEMIEUX.ECCBA.RELEASE]ECCBA3.OBJ;9	6-Sep-1985 17:18	VAX-11 Bliss-32 V4.1-746
UBI_TESTS_6_10	01-04	3529	[LEMIEUX.ECCBA.RELEASE]ECCBA4.OBJ;8	6-Sep-1985 17:19	VAX-11 Bliss-32 V4.1-746
UBI_TESTS_11_15	01-04	4251	[LEMIEUX.ECCBA.RELEASE]ECCBA5.OBJ;8	6-Sep-1985 17:19	VAX-11 Bliss-32 V4.1-746
UBI_TESTS_16_22	01-04	4893	[LEMIEUX.ECCBA.RELEASE]ECCBA6.OBJ;7	6-Sep-1985 17:20	VAX-11 Bliss-32 V4.1-746
UBI_TESTS_23_26	01-04	2843	[LEMIEUX.ECCBA.RELEASE]ECCBA7.OBJ;7	6-Sep-1985 17:24	VAX-11 Bliss-32 V4.1-746
UBI_TESTS_27_32	01-04	8641	[LEMIEUX.ECCBA.RELEASE]ECCBA8.OBJ;16	6-Sep-1985 17:25	VAX-11 Bliss-32 V4.1-746
DS	.STB;2	0	[SOURCE.GENERIC]DS.STB;1	17-NOV-1980 11:19	LINK-32 V2B.44

+-----+
! Program Section Synopsis !
+-----+

Psect Name	Module Name	Base	End	Length	Align	Attributes
\$HEADER	UBI_HEADER	00000200	00000257	00000058 (	88.) PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
\$OWN\$	UBI_TESTS_1_5	00000258	000002E9	00000092 (	146.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
	UBI_TESTS_6_10	00000258	0000026D	00000016 (	22.) LONG 2	
	UBI_TESTS_11_15	00000270	00000285	00000016 (	22.) LONG 2	
	UBI_TESTS_16_22	00000288	000002A1	0000001A (	26.) LONG 2	
	UBI_TESTS_23_26	000002A4	000002B9	00000016 (	22.) LONG 2	
	UBI_TESTS_27_32	000002BC	000002D1	00000016 (	22.) LONG 2	
\$PLIT\$	UBI_TESTS_27_32	000002D4	000002E9	00000016 (	22.) LONG 2	
	UBI_HEADER	000002EC	0000151F	00001234 (	4660.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	UBI_TESTS_1_5	000002EC	00001109	00000E1E (	3614.) LONG 2	
	UBI_TESTS_6_10	0000110C	0000119B	00000090 (	144.) LONG 2	
	UBI_TESTS_11_15	0000119C	0000123F	000000A4 (	164.) LONG 2	
	UBI_TESTS_16_22	00001240	000012FB	000000BC (	188.) LONG 2	
	UBI_TESTS_23_26	000012FC	000013EB	000000F0 (	240.) LONG 2	
	UBI_TESTS_27_32	000013EC	0000146F	00000084 (	132.) LONG 2	
\$TSTCNT	UBI_TESTS_27_32	00001470	0000151F	000000B0 (	176.) LONG 2	
	UBI_HEADER	00001520	00001523	00000004 (	4.) LONG 2	NOPIC,USR,OV,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	UBI_SUBROUTINES	00001520	00001523	00000004 (	4.) LONG 2	
	UBI_TESTS_1_5	00001520	00001523	00000004 (	4.) LONG 2	
	UBI_TESTS_6_10	00001520	00001523	00000004 (	4.) LONG 2	
	UBI_TESTS_11_15	00001520	00001523	00000004 (	4.) LONG 2	
	UBI_TESTS_16_22	00001520	00001523	00000004 (	4.) LONG 2	
	UBI_TESTS_27_32	00001520	00001523	00000004 (	4.) LONG 2	
. BLANK .	UBI_ISR	00001524	00001527	00000004 (	4.) BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
CLEANUP	UBI_HEADER	00001528	0000155F	00000038 (	56.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
CODE	UBI_HEADER	00001560	00001BA7	00000648 (	1608.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC

Psect Name	Module Name	Base	End	Length	Align	Attributes
CODE		00001560	00001BA7	00000648 (	1608.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
DISPATCH	UBI_SUBROUTINES	00001560	00001BA7	00000648 (	1608.) LONG 2	
		00001BA8	00001EA7	00000300 (	768.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	UBI_TESTS_1_5	00001BA8	00001C1F	00000078 (	120.) LONG 2	
	UBI_TESTS_6_10	00001C20	00001C97	00000078 (	120.) LONG 2	
	UBI_TESTS_11_15	00001C98	00001D0F	00000078 (	120.) LONG 2	
	UBI_TESTS_16_22	00001D10	00001DB7	000000A8 (	168.) LONG 2	
	UBI_TESTS_23_26	00001DB8	00001E17	00000060 (	96.) LONG 2	
	UBI_TESTS_27_32	00001E18	00001EA7	00000090 (	144.) LONG 2	
DISPATCH_X		00001EA8	00001EBF	00000018 (	24.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
GLOBALS	UBI_HEADER	00001EA8	00001EBF	00000018 (	24.) LONG 2	
		00001EC0	00002429	0000056A (	1386.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
INITIALIZE	UBI_HEADER	00001EC0	00002429	0000056A (	1386.) LONG 2	
		0000242C	00002569	0000013E (	318.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
SCRATCHPAD	UBI_HEADER	0000242C	00002569	0000013E (	318.) LONG 2	
		00002600	000027FF	00000200 (	512.) PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
SUMMARY	UBI_HEADER	00002600	000027FF	00000200 (	512.) PAGE 9	
		00002800	00002809	0000000A (	10.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
TEST_001	UBI_HEADER	00002800	00002809	0000000A (	10.) LONG 2	
		0000280C	000028AE	000000A3 (	163.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_002	UBI_TESTS_1_5	0000280C	000028AE	000000A3 (	163.) LONG 2	
		000028B0	00002B54	000002A5 (	677.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_003	UBI_TESTS_1_5	000028B0	00002B54	000002A5 (	677.) LONG 2	
		00002B58	00002BE6	0000008F (	143.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_004	UBI_TESTS_1_5	00002B58	00002BE6	0000008F (	143.) LONG 2	
		00002BE8	00002C99	000000B2 (	178.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_005	UBI_TESTS_1_5	00002BE8	00002C99	000000B2 (	178.) LONG 2	
		00002C9C	00002D8B	000000F0 (	240.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_006	UBI_TESTS_1_5	00002C9C	00002D8B	000000F0 (	240.) LONG 2	
		00002D8C	00003003	00000278 (	632.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_007	UBI_TESTS_6_10	00002D8C	00003003	00000278 (	632.) LONG 2	
		00003004	000033EE	000003EB (	1003.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_008	UBI_TESTS_6_10	00003004	000033EE	000003EB (	1003.) LONG 2	
		000033F0	0000351E	0000012F (	303.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_009	UBI_TESTS_6_10	000033F0	0000351E	0000012F (	303.) LONG 2	
		00003520	00003901	000003E2 (	994.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_010	UBI_TESTS_6_10	00003520	00003901	000003E2 (	994.) LONG 2	
		00003904	00003A22	0000011F (	287.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_011	UBI_TESTS_6_10	00003904	00003A22	0000011F (	287.) LONG 2	
		00003A24	00003CF3	000002D0 (	720.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_012	UBI_TESTS_11_15	00003A24	00003CF3	000002D0 (	720.) LONG 2	
		00003CF4	00003E24	00000131 (	305.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_013	UBI_TESTS_11_15	00003CF4	00003E24	00000131 (	305.) LONG 2	
		00003E28	000045A0	00000779 (	1913.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_014	UBI_TESTS_11_15	00003E28	000045A0	00000779 (	1913.) LONG 2	
		000045A4	0000481A	00000277 (	631.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_015	UBI_TESTS_11_15	000045A4	0000481A	00000277 (	631.) LONG 2	
		0000481C	00004973	00000158 (	344.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_016	UBI_TESTS_11_15	0000481C	00004973	00000158 (	344.) LONG 2	
		00004974	00004CB7	00000344 (	836.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC

Psect Name	Module Name	Base	End	Length	Align	Attributes
TEST_016	UBI_TESTS_16_22	00004974	00004CB7	00000344 (	836.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_017	UBI_TESTS_16_22	00004CB8	00005156	0000049F (	1183.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_018	UBI_TESTS_16_22	00005158	000053AA	00000253 (	595.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_019	UBI_TESTS_16_22	000053AC	000055C3	00000218 (	536.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_020	UBI_TESTS_16_22	000055C4	00005728	00000165 (	357.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_021	UBI_TESTS_16_22	0000572C	00005862	00000137 (	311.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_022	UBI_TESTS_16_22	00005864	00005AE4	00000281 (	641.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_023	UBI_TESTS_23_26	00005AE8	00005C2C	00000145 (	325.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_024	UBI_TESTS_23_26	00005C30	00006097	00000468 (	1128.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_025	UBI_TESTS_23_26	00006098	000061D3	0000013C (	316.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_026	UBI_TESTS_23_26	000061D4	0000650B	00000338 (	824.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_027	UBI_TESTS_27_32	0000650C	00006AC5	000005BA (	1466.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_028	UBI_TESTS_27_32	00006AC8	00007113	0000064C (	1612.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_029	UBI_TESTS_27_32	00007114	000074F0	000003DD (	989.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_030	UBI_TESTS_27_32	000074F4	0000756C	00000079 (	121.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_031	UBI_TESTS_27_32	00007570	000075FB	0000008C (	140.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_032	UBI_TESTS_27_32	000075FC	0000857A	00000F7F (	3967.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
UBI_ISR	UBI_ISR	0000857C	00008711	00000196 (	406.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC

! Symbol Cross Reference !

Symbol	Value	Defined By	Referenced By ...
\$ENV	00000001	UBI_ISR	
\$MO	00000001	UBI_ISR	
BUFFER_SETUP	00001956-R	UBI_SUBROUTINES	UBI_TESTS_11_15 UBI_TESTS_23_26
CHAN_STAT	00001EC0-R	UBI_HEADER	UBI_TESTS_16_22 UBI_TESTS_27_32
CLEAN_UP	00001528-R	UBI_HEADER	UBI_TESTS_1_5 UBI_TESTS_6_10
CODEVECTORS	00001EDC-R	UBI_HEADER	UBI_TESTS_23_26 UBI_TESTS_27_32

Symbol	Value	Defined By	Referenced By ...		
CODEWORDS	00001EC8-R	UBI_HEADER	UBI_SUBROUTINES UBI_TESTS_1_5 UBI_TESTS_6_10	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32
CPU\$V_COMET	00000002	DS			
CPU\$V_HS	00000000	DS			
CPU\$V_STAR	00000001	DS			
CSR_MNEM	00000FE6-R	UBI_HEADER			
DECODED_WORDS	00001EF8-R	UBI_HEADER	UBI_SUBROUTINES UBI_TESTS_1_5 UBI_TESTS_6_10	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32
DECODER	00001AE9-R	UBI_SUBROUTINES	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32	UBI_TESTS_1_5 UBI_TESTS_6_10
DEVICES	00001F2A-R	UBI_HEADER			
DS\$ABORT	00010020	DS	UBI_HEADER		
DS\$ASKADR	00010090	DS			
DS\$ASKDATA	00010080	DS			
DS\$ASKLGCL	00010098	DS			
DS\$ASKSTR	000100A0	DS			
DS\$ASKVLD	00010088	DS			
DS\$ATTACH	000101A8	DS			
DS\$BGNSUB	00010030	DS	UBI_TESTS_11_15 UBI_TESTS_6_10	UBI_TESTS_23_26	UBI_TESTS_27_32
DS\$BREAK	00010058	DS	UBI_HEADER UBI_TESTS_1_5 UBI_TESTS_6_10	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32
DS\$CANWAIT	00010070	DS			
DS\$CHANNEL	00010180	DS	UBI_HEADER	UBI_TESTS_27_32	
DS\$CKLOOP	00010040	DS			
DS\$CLRVEC	00010168	DS	UBI_TESTS_27_32		
DS\$CNTRLC	00010078	DS			
DS\$CVTREG	000100B0	DS	UBI_SUBROUTINES		
DS\$ENDPASS	00010010	DS	UBI_HEADER		
DS\$ENDSUB	00010038	DS	UBI_TESTS_11_15 UBI_TESTS_6_10	UBI_TESTS_23_26	UBI_TESTS_27_32
DS\$ERRDEV	000100C8	DS			
DS\$ERRHARD	000100D0	DS	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32	UBI_TESTS_1_5 UBI_TESTS_6_10
DS\$ERRSOFT	000100D8	DS			
DS\$ERRSYS	000100C0	DS	UBI_ISR		
DS\$ESCAPE	00010050	DS			
DS\$GETBUF	00010120	DS	UBI_SUBROUTINES		
DS\$GETMEM	00010130	DS			
DS\$GPHARD	00010018	DS	UBI_HEADER		
DS\$HELP	000101B0	DS			
DS\$INITSCB	00010170	DS	UBI_HEADER		
DS\$INLOOP	00010048	DS	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32	UBI_TESTS_1_5 UBI_TESTS_6_10
DS\$LOAD	00010198	DS			
DS\$MMOFF	00010158	DS			
DS\$MMON	00010150	DS			

Symbol	Value	Defined By	Referenced By ...
DS\$MOVPHY	00010148	DS	
DS\$MOVVRT	00010140	DS	
DS\$PARSE	000100B8	DS	
DS\$PRINTB	000100E0	DS	UBI_SUBROUTINES
DS\$PRINTF	000100F0	DS	UBI_TESTS_27_32
DS\$PRINTS	000100F8	DS	
DS\$PRINTSIG	00010100	DS	
DS\$PRINTX	000100E8	DS	UBI_HEADER
DS\$PROBE	000101A0	DS	UBI_SUBROUTINES
DS\$RELBUF	00010128	DS	UBI_TESTS_27_32
DS\$RELMEM	00010138	DS	
DS\$SETIPL	00010178	DS	UBI_SUBROUTINES
DS\$SETMAP	00010188	DS	UBI_TESTS_27_32
DS\$SETVEC	00010160	DS	UBI_TESTS_6_10
DS\$SHOCHAN	00010190	DS	UBI_HEADER
DS\$SUMMARY	00010028	DS	UBI_TESTS_27_32
DS\$WAITMS	00010060	DS	UBI_TESTS_23_26
DS\$WAITUS	00010068	DS	UBI_TESTS_11_15
			UBI_TESTS_27_32
DSA\$AL_APTMAIL	0000FE00	DS	UBI_TESTS_27_32
DSA\$AT_APTTXT	0000FA00	DS	UBI_TESTS_16_22
DSA\$GL_APTCOM	0000FE04	DS	UBI_TESTS_6_10
DSA\$GL_DEVLEN	0000FE58	DS	
DSA\$GL_ERRNO	0000FE44	DS	
DSA\$GL_EVENT	0000FE48	DS	
DSA\$GL_FLAGS	0000FE00	DS	
DSA\$GL_MSGTYP	0000FE40	DS	
DSA\$GL_PASSES	0000FE08	DS	
DSA\$GL_PASSNO	0000FE54	DS	
DSA\$GL_SECTNO	0000FE10	DS	
DSA\$GL_SID	0000FE14	DS	
DSA\$GL_SUBTNO	0000FE4C	DS	
DSA\$GL_TESTNO	0000FE50	DS	
DSA\$GL_UNITS	0000FE0C	DS	
DSA\$GQ_MSGPTR	0000FE68	DS	
DSA\$GT_DEVNAM	0000FESC	DS	
DSA\$M_APT	80000000	DS	
DSA\$M_BELL	00000008	DS	
DSA\$M_COMPAT	04000000	DS	
DSA\$M_ENSPEC	40000000	DS	
DSA\$M_HALTD	00000001	DS	
DSA\$M_HALTI	00000002	DS	
DSA\$M_IE1	00000010	DS	
DSA\$M_IE2	00000020	DS	
DSA\$M_IE3	00000040	DS	
DSA\$M_IES	00000080	DS	
DSA\$M_LOCK	00000800	DS	
DSA\$M_LOOP	00000004	DS	
DSA\$M_NORPT	08000000	DS	
DSA\$M_OPER	00001000	DS	

Symbol	Value	Defined By	Referenced By ...
-----	-----	-----	-----
DSA\$M_PASSO	20000000	DS	
DSA\$M_PROMPT	00002000	DS	
DSA\$M_QUICK	00000100	DS	
DSA\$M_SEARCH	00004000	DS	
DSA\$M_SPOOL	00000200	DS	
DSA\$M_TRACE	00000400	DS	
DSA\$M_USER	10000000	DS	
DSA\$V_APT	0000001F	DS	
DSA\$V_BELL	00000003	DS	
DSA\$V_COMPAT	0000001A	DS	
DSA\$V_ENSPEC	0000001E	DS	
DSA\$V_HALTD	00000000	DS	
DSA\$V_HALTI	00000001	DS	
DSA\$V_IE1	00000004	DS	
DSA\$V_IE2	00000005	DS	
DSA\$V_IE3	00000006	DS	
DSA\$V_IES	00000007	DS	
DSA\$V_LOCK	0000000B	DS	
DSA\$V_LOOP	00000002	DS	
DSA\$V_NORPT	0000001B	DS	
DSA\$V_OPER	0000000C	DS	
DSA\$V_PASSO	0000001D	DS	
DSA\$V_PROMPT	0000000D	DS	
DSA\$V_QUICK	00000008	DS	
DSA\$V_SEARCH	0000000E	DS	
DSA\$V_SPOOL	00000009	DS	
DSA\$V_TRACE	0000000A	DS	
DSA\$V_USER	0000001C	DS	
ENCODER	00001AAF-R	UBI_SUBROUTINES	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
EVENT_FLAG	00001F1C-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_16_22 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
EXP_MSG	0000035C-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
EXP_REC_XOR	00000E8C-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
EXTERNAL_FLAG	00001F1D-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
FMT_BDP_DATI	0000037C-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
FMT_BDP_DATO	000003BB-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
FMT_BDP_DATO	000003FA-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
FMT_BODY	00000368-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
FMT_BYTE_OFF	0000043A-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
FMT_CMI_UB	0000047C-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
FMT_CPU_RW	000004BE-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32
FMT_CSR_ERR1	000004F9-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_27_32 UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_16_22 UBI_TESTS_27_32

Symbol	Value	Defined By	Referenced By ...		
FMT_CSR_ERR2	00000508-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_DCMF_ERR1	00000645-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_DCMF_ERR1A	000006FB-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_DCMF_ERR2	00000752-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_DDP_DATI	0000051C-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_DDP_DATO	00000551-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_DDP_DATO	000005A1-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_DP_SEL	00000608-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_EXT_PROC	00000778-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_INIT_ACLO	000007CD-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_INTERLOCK	0000079E-R	UBI_HEADER	UBI_TESTS_11_15		
FMT_MAP_ADDR	00000822-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_MAP_BIT_SAO	00000842-R	UBI_HEADER	UBI_TESTS_1_5		
FMT_MAP_BIT_SAI	00000860-R	UBI_HEADER	UBI_TESTS_1_5		
FMT_MAP_CS	000008A6-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_MAP_DB_SAO	000008C3-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_6_10		
FMT_MAP_DB_SAI	000008E4-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_6_10		
FMT_MAP_FAIL	00000904-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_MAP_GEN	0000087D-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_MAP_INV	00000912-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_MCHK	0000092B-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_NOTHING	00000A74-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_NO_ACINT	00000990-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_NO_ACINT1	000009C8-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_NO_INT	00000968-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_NO_MCHK	00000A02-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_NO_UBE	00000A2B-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_NO_UBI	00000A4D-R	UBI_HEADER			
FMT_UAI	00000A7D-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_UBE_DUMP	00000B02-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_UBI_DUMP	00000B85-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_UB_CMI	00000AC0-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_UB_TO	00000D18-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_UET_ACLO1	00000BD1-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UET_INTD	00000C1C-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UET_INTD2	00000C68-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UET_PB	00000CB4-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_UET_STAT	00000CE1-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5

Symbol	Value	Defined By	Referenced By ...		
FMT_UNX_ACINT	00000DA1-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_UNX_ACINT1	00000DD7-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UNX_ACINT2	00000E10-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UNX_INT	00000D43-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UNX_INT1	00000D71-R	UBI_HEADER	UBI_ISR	UBI_TESTS_27_32	
FMT_VECTOR	00000E38-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FREE_MAP	00001F20-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
HANDLER	0000174E-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
INITIALIZE	0000242C-R	UBI_HEADER			
INTERRUPT_EXP	00001F30-R	UBI_HEADER	UBI_ISR	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
INTERRUPT_OCC	00001F31-R	UBI_HEADER	UBI_ISR	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
INT_VECTOR	00001F2C-R	UBI_HEADER	UBI_ISR	UBI_TESTS_23_26	UBI_TESTS_27_32
LN	00001F2B-R	UBI_HEADER			
LUN	00001F32-R	UBI_HEADER	UBI_ISR	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_16_22	UBI_TESTS_1_5	UBI_TESTS_23_26
			UBI_TESTS_27_32	UBI_TESTS_6_10	
MAP_BUFFERS	0000197E-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
MAP_SETUP	0000192D-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
NOISY_WORDS	00001F34-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
			UBI_TESTS_1_5	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_6_10	UBI_TESTS_23_26	UBI_TESTS_27_32
NUM_DATA_ERR	00001F48-R	UBI_HEADER	UBI_TESTS_6_10		
NUM_MAPS	00001F58-R	UBI_HEADER	UBI_TESTS_6_10		
NUM_UBE	00001F5C-R	UBI_HEADER	UBI_TESTS_6_10		
			UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
NUM_UBI	00001F28-R	UBI_HEADER			
PRINT_CSR_ERR	00001560-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
PRINT_DCOMP_ERR	000015E0-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
PRINT_GENERAL	000016FE-R	UBI_SUBROUTINES	UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
			UBI_ISR	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
PRINT_MAP_ERR	000016C2-R	UBI_SUBROUTINES	UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32

+-----+
! Object Module Synopsis !
+-----+

Module Name	Ident	Bytes	File	Creation Date	Creator
UBI_HEADER	01-04	6012	[LEMIEUX.ECCBA.RELEASE]ECCBA0.OBJ;28	6-Sep-1985 17:17	VAX-11 Bliss-32 V4.1-746
UBI_SUBROUTINES	01-04	1612	[LEMIEUX.ECCBA.RELEASE]ECCBA1.OBJ;11	6-Sep-1985 17:18	VAX-11 Bliss-32 V4.1-746
UBI_ISR	01-04	410	[LEMIEUX.ECCBA.RELEASE]ECCBA2.OBJ;12	6-SEP-1985 17:18	VAX/VMS Macro V04-00
UBI_TESTS_1_5	01-04	1691	[LEMIEUX.ECCBA.RELEASE]ECCBA3.OBJ;9	6-Sep-1985 17:18	VAX-11 Bliss-32 V4.1-746
UBI_TESTS_6_10	01-04	3529	[LEMIEUX.ECCBA.RELEASE]ECCBA4.OBJ;8	6-Sep-1985 17:19	VAX-11 Bliss-32 V4.1-746
UBI_TESTS_11_15	01-04	4251	[LEMIEUX.ECCBA.RELEASE]ECCBA5.OBJ;8	6-Sep-1985 17:19	VAX-11 Bliss-32 V4.1-746
UBI_TESTS_16_22	01-04	4893	[LEMIEUX.ECCBA.RELEASE]ECCBA6.OBJ;7	6-Sep-1985 17:20	VAX-11 Bliss-32 V4.1-746
UBI_TESTS_23_26	01-04	2843	[LEMIEUX.ECCBA.RELEASE]ECCBA7.OBJ;7	6-Sep-1985 17:24	VAX-11 Bliss-32 V4.1-746
UBI_TESTS_27_32	01-04	8641	[LEMIEUX.ECCBA.RELEASE]ECCBA8.OBJ;16	6-Sep-1985 17:25	VAX-11 Bliss-32 V4.1-746
DS	.STB;2	0	[SOURCE.GENERIC]DS.STB;1	17-NOV-1980 11:19	LINK-32 V2B.44

+-----+
! Program Section Synopsis !
+-----+

Psect Name	Module Name	Base	End	Length	Align	Attributes
\$HEADER	UBI_HEADER	00000200	00000257	00000058 (	88.) PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
\$OWNS	UBI_TESTS_1_5	00000258	0000026D	00000016 (	22.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
	UBI_TESTS_6_10	00000270	00000285	00000016 (	22.) LONG 2	
	UBI_TESTS_11_15	00000288	000002A1	0000001A (	26.) LONG 2	
	UBI_TESTS_16_22	000002A4	000002B9	00000016 (	22.) LONG 2	
	UBI_TESTS_23_26	000002BC	000002D1	00000016 (	22.) LONG 2	
	UBI_TESTS_27_32	000002D4	000002E9	00000016 (	22.) LONG 2	
\$PLITS	UBI_HEADER	000002EC	0000151F	00001234 (	4660.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	UBI_TESTS_1_5	0000110C	0000119B	00000090 (	144.) LONG 2	
	UBI_TESTS_6_10	0000119C	0000123F	000000A4 (	164.) LONG 2	
	UBI_TESTS_11_15	00001240	000012FB	000000BC (	188.) LONG 2	
	UBI_TESTS_16_22	000012FC	000013EB	000000F0 (	240.) LONG 2	
	UBI_TESTS_23_26	000013EC	0000146F	00000084 (	132.) LONG 2	
	UBI_TESTS_27_32	00001470	0000151F	000000B0 (	176.) LONG 2	
\$TSTCNT	UBI_HEADER	00001520	00001523	00000004 (	4.) LONG 2	NOPIC,USR,OV,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	UBI_SUBROUTINES	00001520	00001523	00000004 (	4.) LONG 2	
	UBI_TESTS_1_5	00001520	00001523	00000004 (	4.) LONG 2	
	UBI_TESTS_6_10	00001520	00001523	00000004 (	4.) LONG 2	
	UBI_TESTS_11_15	00001520	00001523	00000004 (	4.) LONG 2	
	UBI_TESTS_16_22	00001520	00001523	00000004 (	4.) LONG 2	
	UBI_TESTS_27_32	00001520	00001523	00000004 (	4.) LONG 2	
. BLANK .	UBI_ISR	00001524	00001527	00000004 (	4.) BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
CLEANUP	UBI_HEADER	00001528	0000155F	00000038 (	56.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
CODE	UBI_HEADER	00001560	00001BA7	00000648 (	1608.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC

Psect Name	Module Name	Base	End	Length	Align	Attributes
CODE		00001560	00001BA7	00000648 (	1608.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
DISPATCH	UBI_SUBROUTINES	00001560	00001BA7	00000648 (	1608.) LONG 2	
		00001BA8	00001EA7	00000300 (	768.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	UBI_TESTS_1_5	00001BA8	00001C1F	00000078 (	120.) LONG 2	
	UBI_TESTS_6_10	00001C20	00001C97	00000078 (	120.) LONG 2	
	UBI_TESTS_11_15	00001C98	00001D0F	00000078 (	120.) LONG 2	
	UBI_TESTS_16_22	00001D10	00001DB7	000000A8 (	168.) LONG 2	
	UBI_TESTS_23_26	00001DB8	00001E17	00000060 (	96.) LONG 2	
	UBI_TESTS_27_32	00001E18	00001EA7	00000090 (	144.) LONG 2	
DISPATCH_X		00001EA8	00001EBF	00000018 (	24.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
GLOBALS	UBI_HEADER	00001EA8	00001EBF	00000018 (	24.) LONG 2	
		00001EC0	00002429	0000056A (	1386.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
INITIALIZE	UBI_HEADER	00001EC0	00002429	0000056A (	1386.) LONG 2	
		0000242C	00002569	0000013E (	318.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
SCRATCHPAD	UBI_HEADER	0000242C	00002569	0000013E (	318.) LONG 2	
		00002600	000027FF	00000200 (	512.) PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
SUMMARY	UBI_HEADER	00002600	000027FF	00000200 (	512.) PAGE 9	
		00002800	00002809	0000000A (	10.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
TEST_001	UBI_HEADER	00002800	00002809	0000000A (	10.) LONG 2	
		0000280C	000028AE	000000A3 (	163.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_002	UBI_TESTS_1_5	0000280C	000028AE	000000A3 (	163.) LONG 2	
		000028B0	00002B54	000002A5 (	677.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_003	UBI_TESTS_1_5	000028B0	00002B54	000002A5 (	677.) LONG 2	
		00002B58	00002BE6	0000008F (	143.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_004	UBI_TESTS_1_5	00002B58	00002BE6	0000008F (	143.) LONG 2	
		00002BE8	00002C99	000000B2 (	178.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_005	UBI_TESTS_1_5	00002BE8	00002C99	000000B2 (	178.) LONG 2	
		00002C9C	00002D8B	000000F0 (	240.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_006	UBI_TESTS_1_5	00002C9C	00002D8B	000000F0 (	240.) LONG 2	
		00002D8C	00003003	00000278 (	632.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_007	UBI_TESTS_6_10	00002D8C	00003003	00000278 (	632.) LONG 2	
		00003004	000033EE	000003EB (	1003.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_008	UBI_TESTS_6_10	00003004	000033EE	000003EB (	1003.) LONG 2	
		000033F0	0000351E	0000012F (	303.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_009	UBI_TESTS_6_10	000033F0	0000351E	0000012F (	303.) LONG 2	
		00003520	00003901	000003E2 (	994.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_010	UBI_TESTS_6_10	00003520	00003901	000003E2 (	994.) LONG 2	
		00003904	00003A22	0000011F (	287.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_011	UBI_TESTS_6_10	00003904	00003A22	0000011F (	287.) LONG 2	
		00003A24	00003CF3	000002D0 (	720.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_012	UBI_TESTS_11_15	00003A24	00003CF3	000002D0 (	720.) LONG 2	
		00003CF4	00003E24	00000131 (	305.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_013	UBI_TESTS_11_15	00003CF4	00003E24	00000131 (	305.) LONG 2	
		00003E28	000045A0	00000779 (	1913.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_014	UBI_TESTS_11_15	00003E28	000045A0	00000779 (	1913.) LONG 2	
		000045A4	0000481A	00000277 (	631.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_015	UBI_TESTS_11_15	000045A4	0000481A	00000277 (	631.) LONG 2	
		0000481C	00004973	00000158 (	344.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_016	UBI_TESTS_11_15	0000481C	00004973	00000158 (	344.) LONG 2	
		00004974	00004CB7	00000344 (	836.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC

Psect Name	Module Name	Base	End	Length	Align	Attributes
TEST_016	UBI_TESTS_16_22	00004974	00004CB7	00000344 (	836.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_017	UBI_TESTS_16_22	00004974	00004CB7	00000344 (	836.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_018	UBI_TESTS_16_22	00004CB8	00005156	0000049F (	1183.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_019	UBI_TESTS_16_22	00004CB8	00005156	0000049F (	1183.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_020	UBI_TESTS_16_22	00005158	000053AA	00000253 (	595.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_021	UBI_TESTS_16_22	00005158	000053AA	00000253 (	595.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_022	UBI_TESTS_16_22	000053AC	000055C3	00000218 (	536.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_023	UBI_TESTS_16_22	000053AC	000055C3	00000218 (	536.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_024	UBI_TESTS_16_22	000055C4	00005728	00000165 (	357.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_025	UBI_TESTS_16_22	000055C4	00005728	00000165 (	357.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_026	UBI_TESTS_16_22	0000572C	00005862	00000137 (	311.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_027	UBI_TESTS_16_22	0000572C	00005862	00000137 (	311.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_028	UBI_TESTS_16_22	00005864	00005AE4	00000281 (	641.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_029	UBI_TESTS_16_22	00005864	00005AE4	00000281 (	641.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_030	UBI_TESTS_23_26	00005AE8	00005C2C	00000145 (	325.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_031	UBI_TESTS_23_26	00005AE8	00005C2C	00000145 (	325.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
TEST_032	UBI_TESTS_23_26	00005C30	00006097	00000468 (	1128.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
UBI_ISR	UBI_TESTS_23_26	00005C30	00006097	00000468 (	1128.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_23_26	00006098	000061D3	0000013C (	316.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_23_26	00006098	000061D3	0000013C (	316.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_23_26	000061D4	0000650B	00000338 (	824.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_23_26	000061D4	0000650B	00000338 (	824.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	0000650C	00006AC5	000005BA (	1466.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	0000650C	00006AC5	000005BA (	1466.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	00006AC8	00007113	0000064C (	1612.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	00006AC8	00007113	0000064C (	1612.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	00007114	000074F0	000003DD (	989.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	00007114	000074F0	000003DD (	989.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	000074F4	0000756C	00000079 (	121.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	000074F4	0000756C	00000079 (	121.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	00007570	000075FB	0000008C (	140.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	00007570	000075FB	0000008C (	140.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	000075FC	0000857A	00000F7F (	3967.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_TESTS_27_32	000075FC	0000857A	00000F7F (	3967.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_ISR	0000857C	00008711	00000196 (	406.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	UBI_ISR	0000857C	00008711	00000196 (	406.) LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC

! Symbol Cross Reference !

Symbol	Value	Defined By	Referenced By ...
\$ENV	00000001	UBI_ISR	
\$MO	00000001	UBI_ISR	
BUFFER_SETUP	00001956-R	UBI_SUBROUTINES	UBI_TESTS_11_15 UBI_TESTS_23_26
CHAN_STAT	00001EC0-R	UBI_HEADER	UBI_TESTS_16_22 UBI_TESTS_27_32
CLEAN_UP	00001528-R	UBI_HEADER	UBI_TESTS_23_26
CODEVECTORS	00001EDC-R	UBI_HEADER	UBI_TESTS_1_5 UBI_TESTS_6_10 UBI_TESTS_27_32

Symbol	Value	Defined By	Referenced By ...		
CODEWORDS	00001EC8-R	UBI_HEADER	UBI_SUBROUTINES UBI_TESTS_1_5 UBI_TESTS_6_10	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32
CPU\$V_COMET	00000002	DS			
CPU\$V_HS	00000000	DS			
CPU\$V_STAR	00000001	DS			
CSR_MNEM	00000FE6-R	UBI_HEADER	UBI_SUBROUTINES		
DECODED_WORDS	00001EF8-R	UBI_HEADER	UBI_SUBROUTINES UBI_TESTS_1_5 UBI_TESTS_6_10	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32
DECODER	00001AE9-R	UBI_SUBROUTINES	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32	UBI_TESTS_1_5 UBI_TESTS_6_10
DEVICES	00001F2A-R	UBI_HEADER			
DS\$ABORT	00010020	DS	UBI_HEADER		
DS\$ASKADR	00010090	DS			
DS\$ASKDATA	00010080	DS			
DS\$ASKLGCL	00010098	DS			
DS\$ASKSTR	000100A0	DS			
DS\$ASKVLD	00010088	DS			
DS\$ATTACH	000101A8	DS			
DS\$BGNSUB	00010030	DS	UBI_TESTS_11_15 UBI_TESTS_6_10	UBI_TESTS_23_26	UBI_TESTS_27_32
DS\$BREAK	00010058	DS	UBI_HEADER UBI_TESTS_1_5 UBI_TESTS_6_10	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32
DS\$CANWAIT	00010070	DS			
DS\$CHANNEL	00010180	DS	UBI_HEADER	UBI_TESTS_27_32	
DS\$CKLOOP	00010040	DS			
DS\$CLRVEC	00010168	DS	UBI_TESTS_27_32		
DS\$CNTRLC	00010078	DS			
DS\$CVTREG	000100B0	DS	UBI_SUBROUTINES		
DS\$ENDPASS	00010010	DS	UBI_HEADER		
DS\$ENDSUB	00010038	DS	UBI_TESTS_11_15 UBI_TESTS_6_10	UBI_TESTS_23_26	UBI_TESTS_27_32
DS\$ERRDEV	000100C8	DS			
DS\$ERRHARD	000100D0	DS	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32	UBI_TESTS_1_5 UBI_TESTS_6_10
DS\$ERRSOFT	000100D8	DS			
DS\$ERRSYS	000100C0	DS	UBI_ISR		
DS\$ESCAPE	00010050	DS			
DS\$GETBUF	00010120	DS	UBI_SUBROUTINES		
DS\$GETMEM	00010130	DS			
DS\$GPARD	00010018	DS	UBI_HEADER		
DS\$HELP	000101B0	DS			
DS\$INITSCB	00010170	DS	UBI_HEADER		
DS\$INLOOP	00010048	DS	UBI_TESTS_11_15 UBI_TESTS_23_26	UBI_TESTS_16_22 UBI_TESTS_27_32	UBI_TESTS_1_5 UBI_TESTS_6_10
DS\$LOAD	00010198	DS			
DS\$MMOFF	00010158	DS			
DS\$MMON	00010150	DS			

Symbol	Value	Defined By	Referenced By ...
-----	-----	-----	-----
DS\$MOVPHY	00010148	DS	
DS\$MOVVRT	00010140	DS	
DS\$PARSE	000100B8	DS	
DS\$PRINTB	000100E0	DS	UBI_SUBROUTINES
DS\$PRINTF	000100F0	DS	UBI_TESTS_27_32
DS\$PRINTS	000100F8	DS	
DS\$PRINTSIG	00010100	DS	
DS\$PRINTX	000100E8	DS	UBI_HEADER
DS\$PROBE	000101A0	DS	UBI_SUBROUTINES
DS\$RELBUF	00010128	DS	UBI_TESTS_27_32
DS\$RELMEM	00010138	DS	
DS\$SETIPL	00010178	DS	UBI_SUBROUTINES
DS\$SETMAP	00010188	DS	UBI_TESTS_27_32
DS\$SETVEC	00010160	DS	UBI_TESTS_27_32
DS\$SHOCHAN	00010190	DS	
DS\$SUMMARY	00010028	DS	
DS\$WAITMS	00010060	DS	UBI_TESTS_23_26
DS\$WAITUS	00010068	DS	UBI_TESTS_27_32
DS\$AL_APTMAIL	0000FE00	DS	UBI_TESTS_11_15
DS\$AT_APTTXT	0000FA00	DS	UBI_TESTS_16_22
DS\$GL_APTCOM	0000FE04	DS	UBI_TESTS_6_10
DS\$GL_DEVLEN	0000FE58	DS	
DS\$GL_ERRNO	0000FE44	DS	
DS\$GL_EVENT	0000FE48	DS	
DS\$GL_FLAGS	0000FE00	DS	
DS\$GL_MSGTYP	0000FE40	DS	
DS\$GL_PASSES	0000FE08	DS	
DS\$GL_PASSNO	0000FE54	DS	
DS\$GL_SECTNO	0000FE10	DS	
DS\$GL_SID	0000FE14	DS	
DS\$GL_SUBTNO	0000FE4C	DS	
DS\$GL_TESTNO	0000FE50	DS	
DS\$GL_UNITS	0000FE0C	DS	
DS\$GQ_MSGPTR	0000FE68	DS	
DS\$GT_DEVNAM	0000FE5C	DS	
DS\$M_APT	80000000	DS	
DS\$M_BELL	00000008	DS	
DS\$M_COMPAT	04000000	DS	
DS\$M_ENSPEC	40000000	DS	
DS\$M_HALTD	00000001	DS	
DS\$M_HALTI	00000002	DS	
DS\$M_IE1	00000010	DS	
DS\$M_IE2	00000020	DS	
DS\$M_IE3	00000040	DS	
DS\$M_IES	00000080	DS	
DS\$M_LOCK	00000800	DS	
DS\$M_LOOP	00000004	DS	
DS\$M_NORPT	08000000	DS	
DS\$M_OPER	00001000	DS	

Symbol	Value	Defined By	Referenced By ...
DS\$M_PASSO	20000000	DS	
DS\$M_PROMPT	00002000	DS	
DS\$M_QUICK	00000100	DS	
DS\$M_SEARCH	00004000	DS	
DS\$M_SPOOL	00000200	DS	
DS\$M_TRACE	00000400	DS	
DS\$M_USER	10000000	DS	
DS\$V_APT	0000001F	DS	
DS\$V_BELL	00000003	DS	
DS\$V_COMPAT	0000001A	DS	
DS\$V_ENSPEC	0000001E	DS	
DS\$V_HALTD	00000000	DS	
DS\$V_HALTI	00000001	DS	
DS\$V_IE1	00000004	DS	
DS\$V_IE2	00000005	DS	
DS\$V_IE3	00000006	DS	
DS\$V_IES	00000007	DS	
DS\$V_LOCK	0000000B	DS	
DS\$V_LOOP	00000002	DS	
DS\$V_NORPT	0000001B	DS	
DS\$V_OPER	0000000C	DS	
DS\$V_PASSO	0000001D	DS	
DS\$V_PROMPT	0000000D	DS	
DS\$V_QUICK	00000008	DS	
DS\$V_SEARCH	0000000E	DS	
DS\$V_SPOOL	00000009	DS	
DS\$V_TRACE	0000000A	DS	
DS\$V_USER	0000001C	DS	
ENCODER	00001AAF-R	UBI_SUBROUTINES	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
EVENT_FLAG	00001F1C-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_16_22 UBI_TESTS_27_32
EXP_MSG	0000035C-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
EXP_REC_XOR	00000E8C-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
EXTERNAL_FLAG	00001F1D-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
FMT_BDP_DATI	0000037C-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
FMT_BDP_DATO	000003BB-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
FMT_BDP_DATOB	000003FA-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
FMT_BODY	00000368-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
FMT_BYTE_OFF	0000043A-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
FMT_CMI_UB	0000047C-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
FMT_CPU_RW	000004BE-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10
FMT_CSR_ERR1	000004F9-R	UBI_HEADER	UBI_TESTS_11_15 UBI_TESTS_23_26 UBI_TESTS_16_22 UBI_TESTS_1_5 UBI_TESTS_6_10

Symbol	Value	Defined By	Referenced By ...		
FMT_CSR_ERR2	00000508-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_DCMP_ERR1	00000645-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_DCMP_ERR1A	000006FB-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_DCMP_ERR2	00000752-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_DDP_DATI	0000051C-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_DDP_DATO	00000551-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_DDP_DATO	000005A1-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_DP_SEL	00000608-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_EXT_PROC	00000778-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_INIT_ACLO	000007CD-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_INTERLOCK	0000079E-R	UBI_HEADER	UBI_TESTS_11_15		
FMT_MAP_ADDR	00000822-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_MAP_BIT_SA0	00000842-R	UBI_HEADER	UBI_TESTS_1_5		
FMT_MAP_BIT_SA1	00000860-R	UBI_HEADER	UBI_TESTS_1_5		
FMT_MAP_CS	000008A6-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_MAP_DB_SA0	000008C3-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_6_10		
FMT_MAP_DB_SA1	000008E4-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_6_10		
FMT_MAP_FAIL	00000904-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_MAP_GEN	0000087D-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_MAP_INV	00000912-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_MCHK	0000092B-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_NOTHING	00000A74-R	UBI_HEADER	UBI_SUBROUTINES		
FMT_NO_ACINT	00000990-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_NO_ACINT1	000009C8-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_NO_INT	00000968-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_NO_MCHK	00000A02-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_NO_UBE	00000A2B-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_NO_UBI	00000A4D-R	UBI_HEADER			
FMT_UAI	00000A7D-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_UBE_DUMP	00000B02-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_UBI_DUMP	00000B85-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_UB_CMI	00000AC0-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_UB_TO	00000D18-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_UET_ACLO1	00000BD1-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UET_INTD	00000C1C-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UET_INTD2	00000C68-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UET_PB	00000CB4-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FMT_UET_STAT	00000CE1-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5

Symbol	Value	Defined By	Referenced By ...		
FMT_UNX_ACINT	00000DA1-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
FMT_UNX_ACINT1	00000DD7-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UNX_ACINT2	00000E10-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UNX_INT	00000D43-R	UBI_HEADER	UBI_TESTS_27_32		
FMT_UNX_INT1	00000D71-R	UBI_HEADER	UBI_ISR	UBI_TESTS_27_32	
FMT_VECTOR	00000E38-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
FREE_MAP	00001F20-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
HANDLER	0000174E-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
INITIALIZE	0000242C-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
INTERRUPT_EXP	00001F30-R	UBI_HEADER	UBI_ISR	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
INTERRUPT_OCC	00001F31-R	UBI_HEADER	UBI_ISR	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
INT_VECTOR	00001F2C-R	UBI_HEADER	UBI_ISR	UBI_TESTS_23_26	UBI_TESTS_27_32
LN	00001F2B-R	UBI_HEADER			
LUN	00001F32-R	UBI_HEADER	UBI_ISR	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_16_22	UBI_TESTS_1_5	UBI_TESTS_23_26
			UBI_TESTS_27_32	UBI_TESTS_6_10	
MAP_BUFFERS	0000197E-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
MAP_SETUP	0000192D-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
NOISY_WORDS	00001F34-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
NUM_DATA_ERR	00001F48-R	UBI_HEADER	UBI_TESTS_6_10		
NUM_MAPS	00001F58-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
NUM_UBE	00001F5C-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
			UBI_TESTS_11_15		
NUM_UBI	00001F28-R	UBI_HEADER			
PRINT_CSR_ERR	00001560-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
PRINT_DCMP_ERR	000015E0-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
PRINT_GENERAL	000016FE-R	UBI_SUBROUTINES	UBI_TESTS_23_26	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_ISR	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_1_5		
			UBI_TESTS_6_10		
PRINT_MAP_ERR	000016C2-R	UBI_SUBROUTINES	UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
PROBE_ADDRESS	00001841-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
PTBASE	00001F60-R	UBI_HEADER			
REC_MSG	00000E79-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
SCRATCH	00002600-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10

Symbol	Value	Defined By	Referenced By ...
-----	-----	-----	-----
SUMMARY	00002800-R	UBI_HEADER	
SYS\$ALLOC	00010238	DS	
SYS\$ASCTIM	00010248	DS	
SYS\$ASSIGN	00010250	DS	
SYS\$BINTIM	00010258	DS	
SYS\$CANCEL	00010260	DS	
SYS\$CANTIM	00010268	DS	
SYS\$CLOSE	000105B8	DS	
SYS\$CLREF	00010298	DS	
SYS\$CONNECT	000105C0	DS	
SYS\$DALLOC	000102D8	DS	
SYS\$DASSGN	000102E0	DS	
SYS\$DISCONNECT	000105D0	DS	
SYS\$FA0	00010350	DS	
SYS\$FA0L	00010358	DS	
SYS\$GET	00010580	DS	
SYS\$GETCHN	000104C8	DS	
SYS\$GETTIM	00010378	DS	
SYS\$LKWSET	000103A0	DS	
SYS\$NUMTIM	000103B8	DS	
SYS\$OPEN	00010608	DS	
SYS\$QIO	000103C8	DS	
SYS\$QIOW	00010200	DS	
SYS\$READ	00010590	DS	
SYS\$READEF	000103D0	DS	
SYS\$SETAST	000103F8	DS	
SYS\$SETEF	00010400	DS	
SYS\$SETIMR	00010420	DS	
SYS\$SETPRI	00010428	DS	
SYS\$SETPRT	00010430	DS	
SYS\$SETRWM	00010438	DS	
SYS\$SETSWM	00010448	DS	
SYS\$ULKPAG	00010460	DS	
SYS\$ULWSET	00010468	DS	
SYS\$UNWIND	00010470	DS	
SYS\$WAITFR	00010478	DS	
SYS\$WFLAND	00010488	DS	
SYS\$WFLOR	00010490	DS	
TEST_001	0000280C-R	UBI_TESTS_1_5	
TEST_002	000028B0-R	UBI_TESTS_1_5	
TEST_003	00002B58-R	UBI_TESTS_1_5	
TEST_004	00002BE8-R	UBI_TESTS_1_5	
TEST_005	00002C9C-R	UBI_TESTS_1_5	
TEST_006	00002D8C-R	UBI_TESTS_6_10	
TEST_007	00003004-R	UBI_TESTS_6_10	
TEST_008	000033F0-R	UBI_TESTS_6_10	
TEST_009	00003520-R	UBI_TESTS_6_10	
TEST_010	00003904-R	UBI_TESTS_6_10	
TEST_011	00003A24-R	UBI_TESTS_11_15	
TEST_012	00003CF4-R	UBI_TESTS_11_15	

UBI_SUBROUTINES

Symbol	Value	Defined By	Referenced By ...		
TEST_013	00003E28-R	UBI_TESTS_11_15			
TEST_014	000045A4-R	UBI_TESTS_11_15			
TEST_015	0000481C-R	UBI_TESTS_11_15			
TEST_016	00004974-R	UBI_TESTS_16_22			
TEST_017	00004CB8-R	UBI_TESTS_16_22			
TEST_018	00005158-R	UBI_TESTS_16_22			
TEST_019	000053AC-R	UBI_TESTS_16_22			
TEST_020	000055C4-R	UBI_TESTS_16_22			
TEST_021	0000572C-R	UBI_TESTS_16_22			
TEST_022	00005864-R	UBI_TESTS_16_22			
TEST_023	00005AE8-R	UBI_TESTS_23_26			
TEST_024	00005C30-R	UBI_TESTS_23_26			
TEST_025	00006098-R	UBI_TESTS_23_26			
TEST_026	000061D4-R	UBI_TESTS_23_26			
TEST_027	0000650C-R	UBI_TESTS_27_32			
TEST_028	00006AC8-R	UBI_TESTS_27_32			
TEST_029	00007114-R	UBI_TESTS_27_32			
TEST_030	000074F4-R	UBI_TESTS_27_32			
TEST_031	00007570-R	UBI_TESTS_27_32			
TEST_032	000075FC-R	UBI_TESTS_27_32			
UBE0_ISR	00008684-R	UBI_ISR	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_11_15
			UBI_TESTS_16_22	UBI_TESTS_1_5	UBI_TESTS_23_26
			UBI_TESTS_27_32	UBI_TESTS_6_10	
UBE1_ISR	000086A8-R	UBI_ISR	UBI_HEADER	UBI_SUBROUTINES	
UBE2_ISR	000086CC-R	UBI_ISR	UBI_HEADER	UBI_SUBROUTINES	
UBE3_ISR	000086F0-R	UBI_ISR	UBI_HEADER	UBI_SUBROUTINES	
UBECR1_MNEM	00001007-R	UBI_HEADER			
UBECR2_MNEM	00001064-R	UBI_HEADER			
UBE_BASE_ADDR	00001F68-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
UBE_BUF_ADDR	00001F78-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_27_32	
UBE_BUF_SIZE	00001F98-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
UBE_DP_NO	00001F94-R	UBI_HEADER	UBI_ISR	UBI_SUBROUTINES	UBI_TESTS_23_26
			UBI_TESTS_27_32		
UBE_DRIVE	000023E4-R	UBI_HEADER	UBI_TESTS_27_32		
UBE_ERR_BUF	00001FA0-R	UBI_HEADER	UBI_SUBROUTINES		
UBE_EXP_DATA	000023A8-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_23_26	UBI_TESTS_27_32
UBE_INIT_DATA	000023A0-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_23_26	UBI_TESTS_27_32
UBE_INTERRUPT	0000177A-R	UBI_SUBROUTINES	UBI_ISR		
UBE_INT_OCC	000023B0-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_23_26	UBI_TESTS_27_32
UBE_MAP_NO	00001F8C-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_23_26	UBI_TESTS_27_32
UBE_MODE	00001F88-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_23_26	UBI_TESTS_27_32
UBE_PAGE_CNT	00001F9C-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_27_32	
UBE_SEL	00001F29-R	UBI_HEADER			
UBE_SERVICE	000023B4-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
UBE_STAT_ERR	000023D8-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32

Symbol	Value	Defined By	Referenced By ...		
UBE_VECTOR	000023DC-R	UBI_HEADER	UBI_TESTS_6_10	UBI_TESTS_27_32	
UBE_XFER_ERR	000023C4-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_23_26	UBI_TESTS_27_32
UBE_XFER_SIZE	000023C8-R	UBI_HEADER	UBI_SUBROUTINES		
UBIMOD	00000EC6-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_23_26	UBI_TESTS_27_32
			UBI_TESTS_6_10		
UBIMOD_BDP	00000EDA-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_BYTE_OFF	00000EEE-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_CMI	00000EFB-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_CPU_RW	00000F0F-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_CSR	00000F23-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_DDP	00000F3C-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_DP_SEL	00000F50-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_MAP	00000F64-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_MAP_ADDR	00000F7A-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_MAP_CS	00000F8E-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_MAP_CTRL	00000FA2-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_UA1	00000FB6-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBIMOD_UB_CMI	00000FC4-R	UBI_HEADER	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UBI_LINK	000023E8-R	UBI_HEADER	UBI_TESTS_27_32		
UBI_LN	00001EEC-R	UBI_HEADER			
UBI_UNDER_TEST	00001EE4-R	UBI_HEADER	UBI_ISR	UBI_SUBROUTINES	UBI_TESTS_11_15
			UBI_TESTS_16_22	UBI_TESTS_1_5	UBI_TESTS_23_26
			UBI_TESTS_27_32		
UBI_VECTOR	00001EE8-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	
UB_MAP	00001F64-R	UBI_HEADER			
UETMOD	00000FD2-R	UBI_HEADER	UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UET_ISR	0000857C-R	UBI_ISR	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UET_ISR_2	00008600-R	UBI_ISR	UBI_TESTS_27_32		
UNIBUS_SIZER	00001A44-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
UNIBUS_SPACE	00001EF4-R	UBI_HEADER	UBI_ISR	UBI_SUBROUTINES	UBI_TESTS_11_15
			UBI_TESTS_16_22	UBI_TESTS_1_5	UBI_TESTS_23_26
			UBI_TESTS_27_32		
VALID_MAPS	000023EC-R	UBI_HEADER	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22
			UBI_TESTS_1_5	UBI_TESTS_23_26	UBI_TESTS_27_32

Symbol -----	Value -----	Defined By -----	Referenced By ... -----		
			UBI_TESTS_6_10		
XFER_ANALYSIS	000017B0-R	UBI_SUBROUTINES			
XFER_SETUP	00001878-R	UBI_SUBROUTINES	UBI_TESTS_11_15	UBI_TESTS_16_22	UBI_TESTS_1_5
			UBI_TESTS_23_26	UBI_TESTS_27_32	UBI_TESTS_6_10
XOR_MSG	00000E85-R	UBI_HEADER			

Key for special characters above:

```

+-----+
| * - Undefined |
| U - Universal  |
| R - Relocatable|
| X - External   |
+-----+
  
```

! Image Synopsis !

Virtual memory allocated: 00000200 000087FF 00008600 (34304. bytes, 67. pages)
Stack size: 0. pages
Image binary virtual block limits: 1. 67. (67. blocks)
Image name and identification: ECCBA 01-04
Number of files: 10.
Number of modules: 10.
Number of program sections: 49.
Number of global symbols: 325.
Number of cross references: 1406.
Number of image sections: 1.
Image type: SYSTEM.
Map format: DEFAULT WITH CROSS REFERENCE in file DISK\$DIAGPACK04:[LEMIEUX.ECCBA.RELEASE]ECCBA.
Estimated map length: 148. blocks

! Link Run Statistics !

Performance Indicators	Page Faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Command processing:	90	00:00:00.84	00:00:01.53
Pass 1:	116	00:00:01.28	00:00:02.33
Allocation/Relocation:	24	00:00:00.17	00:00:00.64
Pass 2:	99	00:00:01.87	00:00:03.29
Map data after object module synopsis:	4	00:00:01.29	00:00:01.43
Symbol table output:	0	00:00:00.02	00:00:00.09
Total run values:	333	00:00:05.47	00:00:09.31

Using a working set limited to 962 pages and 78 pages of data storage (excluding image)

Total number object records read (both passes): 1477
of which 0 were in libraries and 150 were DEBUG data records containing 11105 bytes

Number of modules extracted explicitly = 0
with 0 extracted to resolve undefined symbols

0 library searches were for symbols not in the library searched

A total of 0 global symbol table records was written

LINK/SYST=%X200/CROSS/EXE=ECCBA/MAP=ECCBA ECCBA0+ECCBA1+ECCBA2+ECCBA3+ECCBA4+ECCBA5+ECCBA6+ECCBA7+ECCBA8+RELD\$0:[SOURCE.GENERIC]DS.S
TB

```

: 0001 0  MODULE UBI_HEADER (      ! Header module of UBI Diagnostic Program
: 0002 0      IDENT = '01-04'
: 0003 0      ) =
: 0004 1  BEGIN
: 0005 1
: 0006 1  !
: 0007 1  ! COPYRIGHT (C) 1980,1981
: 0008 1  ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
: 0009 1  !
: 0010 1  ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
: 0011 1  ! COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
: 0012 1  ! ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
: 0013 1  ! MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
: 0014 1  ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
: 0015 1  ! TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
: 0016 1  ! REMAIN IN DEC.
: 0017 1  !
: 0018 1  ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
: 0019 1  ! AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
: 0020 1  ! CORPORATION.
: 0021 1  !
: 0022 1  ! DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
: 0023 1  ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
: 0024 1  !
: 0025 1
: 0026 1  !**
: 0027 1  ! FACILITY:      VAX-11 Diagnostic Library
: 0028 1
: 0029 1  ! ABSTRACT:      This module contains the global data, ASCII messages,
: 0030 1  !                      initialization code, cleanup code, and summary code for the
: 0031 1  !                      COMET UBI Diagnostic Program.
: 0032 1
: 0033 1  ! ENVIRONMENT:      VAX-11 Diagnostic Supervisor
: 0034 1
: 0035 1  ! AUTHOR:           Rand Gray, CREATION DATE: 28-Nov-78
: 0036 1
: 0037 1  ! MODIFIED BY:      Don Monroe : May 1979
: 0038 1
: 0039 1  !           0-7      May 1979
: 0040 1  !                      Changed references to the RCS module to UBI.
: 0041 1
: 0042 1  !           0-8      July 1979
: 0043 1  !                      Modified to support the real UET module.
: 0044 1
: 0045 1  !           0-9      July 1979
: 0046 1  !                      Changed references to 'Byte Offset' to 'Byte Number'
: 0047 1
: 0048 1  !           0-10     September 1979
: 0049 1  !                      Added purge inside scope loops in module 5.
: 0050 1  !                      Fixed a bug in test 3. It was using map bit 15.
: 0051 1  !                      Fixed bug in INIT code to init the LUN to zero at end of pass
: 0052 1  !                      instead of minus 1.
: 0053 1
: 0054 1  !           0-11     October 1979
: 0055 1  !                      Reference Module 7 Version 10.

```


:	0056	1	:		Added a subtest to DDP DATIP test. Reference module 5.
:	0057	1	:		
:	0058	1	:	0-12	October 16, 1979
:	0059	1	:		Fixed a bug in test 21. Added code to all DDP DATO and DATOB
:	0060	1	:		tests to check entire long words.
:	0061	1	:		
:	0062	1	:	0-13	October 22, 1979
:	0063	1	:		Added typeout of SA0/SA1 and map number/map address to
:	0064	1	:		map entry test. Also added typeout of data.
:	0065	1	:		Added a subroutine to print MAP errors. The subroutine is
:	0066	1	:		invoked thru a print link in the ERRHARD call.
:	0067	1	:		
:	0068	1	:	0-14	December 7, 1979
:	0069	1	:		Fixed bug in Test 6 Subtest 4 to mask the INIT bit in the
:	0070	1	:		UET CTRL register.
:	0071	1	:		Fixed bug in PROBE_ADDRESS routine in the ADAWI builtin.
:	0072	1	:		
:	0073	1	:	0-15	March 24, 1980
:	0074	1	:		Added UBE exerciser test to default section.
:	0075	1	:		
:	0076	1	:	0-16	April 7, 1980
:	0077	1	:		Fixed references to unibus space so that second unibus will
:	0078	1	:		work correctly.
:	0079	1	:		
:	0080	1	:	0-17	April 9, 1980
:	0081	1	:		Added print links to errhards.
:	0082	1	:		Fixed UBI vector for unit 0 and unit 1.
:	0083	1	:		
:	0084	1	:	0-18	April 10, 1980
:	0085	1	:		Updated for version 113 of supervisor.
:	0086	1	:		
:	0087	1	:	0-19	April 14, 1980
:	0088	1	:		Added tests for INTERRUPT DONE in CSR1 for the second unibus
:	0089	1	:		and added test for CSR2 interrupt logic in second unibus.
:	0090	1	:	0-20	April 16, 1980
:	0091	1	:		Added tests for ACLO in CSR2 and CSR1 of the second unibus.
:	0092	1	:	0-21	April 25, 1980
:	0093	1	:		Fixed bug in call to INITSCB.
:	0094	1	:	0-22	April 28, 1980
:	0095	1	:		Fixed bugs in Interrupt Test relating to referancing the
:	0096	1	:		unibus with long data type instead of word data type.
:	0097	1	:	0-23	April 28, 1980
:	0098	1	:		Fixed bug in setting vector in ACLOW interrupt test.
:	0099	1	:	0-24	April 29, 1980
:	0100	1	:		Fixed bug in ACLOW test relating to not waiting for ACLOW
:	0101	1	:		to go away (100 ms).
:	0102	1	:	0-25	May 1, 1980
:	0103	1	:		Fixed bugs in ACLOW test relating to not waiting for UET
:	0104	1	:		INIT to go away and the ACLOW1 error message.
:	0105	1	:	1-00	June 16, 1980
:	0106	1	:		Initial Release
:	0107	1	:	1-01	July 31, 1980
:	0108	1	:		Fixed bugs in UBE test related to 1 memory array.
:	0109	1	:	1-03	September, 1981
:	0110	1	:		Fixed bug in module 1. Also fixed second unibus bugs in module 8.

ZZ-ECCBA-1.4 01-04
UBI_HEADER
01-04

N 4
6-Sep-1985 Fiche 1 Frame N4 Sequence 52
6-Sep-1985 17:17:45 VAX-11 Bliss-32 V4.1-746 Page 3
6-Sep-1985 17:14:42 [LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21 (1)

: 0111 1 ! 'EXPECTED' and 'RECEIVED' was reversed in message 'EXP_REC_XOR'.
: 0112 1 !
: 0113 1 ! Kevin LeMieux
: 0114 1 !
: 0115 1 ! 1-04 February, 1985
: 0116 1 ! Fixed bug in INIT code. If two DW's were specified; after the
: 0117 1 ! first pass one of them one of them was no longer tested.
: 0118 1 ! Fixed minor bugs in TEST 29.
: 0119 1 !--


```

; 0120 1 !
; 0121 1 ! TABLE OF CONTENTS:
; 0122 1 !
; 0123 1 ! INITIALIZE
; 0124 1 ! CLEANUP
; 0125 1 ! SUMMARY
; 0126 1 !
; 0127 1 !
; 0128 1 ! INCLUDE FILES:
; 0129 1 !
; 0130 1
; 0131 1 LIBRARY 'SYS$LIBRARY:DIAG' ;
; 0132 1 LIBRARY 'SYS$LIBRARY:STARLET';
; 0133 1
; 0134 1 LITERAL
; 0135 1 CLEAR_MEM_ERR = %X'E0000000',
; 0136 1 MEMORY_CONT = %X'F20000';
; 0137 1
; 0138 1 !
; 0139 1 ! DIAGNOSTIC SUPERVISOR MACROS:
; 0140 1 !
; 0141 1
; L 0142 1 $DS_BGNMOD(ENV=CEP_REPAIR);
; %PRINT: Bliss-32 Diagnostic Macro Library version 7.0 "DIAG.L32 (57)"
; 0143 1 $DS_BGNREG;
; 0144 1 $DS_ENDREG;
; 0145 1 $DS_DISPATCH;
; 0146 1 $DS_SECTION(ALL,UBE);
; 0147 1 $DS_DEVTYP(STRINGS=(DW750,UBE),ADDRESSES=);
; 0148 1 $DS_DSADEF;
; P 0149 1 $DS_HEADER(PNAME='ECCBA-REV. 1.4 VAX 11/750(UBI), DW750 Diagnostic',REV=01,
; 0150 1 UPDATE=4,NUNIT=10,ERRTYP=0,STAT=0);
; 0151 1
; 0152 1 !
; 0153 1 ! MACROS:
; 0154 1 !
; 0155 1
; 0156 1 MACRO
; 0157 1 ASCIC[] = UPLIT BYTE( %ASCIC %STRING (%REMAINING) )%;
; 0158 1
; 0159 1 !
; 0160 1 ! EQUATED SYMBOLS:
; 0161 1 !

```

Initialization code
Clean up code
Summary reporting code

! A8
! A8

```

: 0162 1 GLOBAL BIND
: 0163 1
: 0164 1 !
: 0165 1 ! ASCII messages
: 0166 1 !
: 0167 1
: P 0168 1 EXP_MSG= ASCII
: 0169 1 (' EXPECTED: '),
: P 0170 1 FMT_BODY= ASCII
: 0171 1 ('!/!AC !XL(X) ; !AC'),
: P 0172 1 FMT_BDP_DATI= ASCII
: 0173 1 ('!/DATA PATH !UB DATI ERROR',
: 0174 1 '!/EXPECTED: !XW(X)',
: 0175 1 '!/RECEIVED: !XW(X)'),
: P 0176 1 FMT_BDP_DATO= ASCII
: 0177 1 ('!/DATA PATH !UB DATO ERROR',
: 0178 1 '!/EXPECTED: !XL(X)',
: 0179 1 '!/RECEIVED: !XL(X)'),
: P 0180 1 FMT_BDP_DATO= ASCII
: 0181 1 ('!/DATA PATH !UB DATO ERROR',
: 0182 1 '!/EXPECTED: !XL(X)',
: 0183 1 '!/RECEIVED: !XL(X)'),
: P 0184 1 FMT_BYTE_OFF= ASCII
: 0185 1 ('!/CMI TO UNIBUS MAPPING ERROR',
: 0186 1 '!/EXPECTED: !XL(X)',
: 0187 1 '!/RECEIVED: !XL(X)'),
: P 0188 1 FMT_CMI_UB= ASCII
: 0189 1 ('!/CMI TO UNIBUS MAPPING ERROR',
: 0190 1 '!/EXPECTED: !XL(X)',
: 0191 1 '!/RECEIVED: !XL(X)'),
: P 0192 1 FMT_CPU_RW= ASCII
: 0193 1 ('!/CPU READ/WRITE ERROR',
: 0194 1 '!/EXPECTED: !XW(X)',
: 0195 1 '!/RECEIVED: !XW(X)'),
: P 0196 1 FMT_CSR_ERR1= ASCII
: 0197 1 ('!/CSR!UB ERROR'),
: P 0198 1 FMT_CSR_ERR2= ASCII
: 0199 1 ('!/(CSR!UB) = !XL(X)'),
: P 0200 1 FMT_DDP_DATI= ASCII
: 0201 1 ('!/DDP DATI ERROR',
: 0202 1 '!/EXPECTED: !XW(X)',
: 0203 1 '!/RECEIVED: !XW(X)'),
: P 0204 1 FMT_DDP_DATO= ASCII
: 0205 1 ('!/DDP DATO ERROR',
: 0206 1 '!/CMI ADDRESS IN ERROR: !XL', ! A12
: 0207 1 '!/EXPECTED: !XW(X)',
: 0208 1 '!/RECEIVED: !XW(X)'),
: P 0209 1 FMT_DDP_DATO= ASCII
: 0210 1 ('!/DDP DATO ERROR',
: 0211 1 '!/CMI ADDRESS IN ERROR: !XL(X)', ! A12
: 0212 1 '!/BYTE WRITTEN: !UB', ! M12
: 0213 1 '!/EXPECTED: !XW(X)',
: 0214 1 '!/RECEIVED: !XW(X)'),
: P 0215 1 FMT_DP_SEL= ASCII
: 0216 1 ('!/DATA PATH SELECT ERROR',

```



```

; P 0217 1      '!/EXPECTED: !XL(X)',
; 0218 1      '!/RECIEVED: !XL(X)',
; P 0219 1      FMT_DCOMP_ERR1= ASCII
; P 0220 1      ('!/DATA COMPARE ERROR',
; P 0221 1      '!/XFER SIZE: !XL(X) BYTES STARTING ADDRESS: !XL(X)',
; P 0222 1      '    TOTAL WORDS BAD: !XL(X)',
; P 0223 1      '!/UBE #: !UB DATA PATH #: !UB (4 = ROTATING)',
; 0224 1      '!/BYTE OFFSET: !UB (0 = NO, 1 = YES)!/' ),
; P 0225 1      FMT_DCOMP_ERR1A= ASCII
; P 0226 1      ('!/ ADDR          GOOD          BAD          XOR',      ! M8
; 0227 1      '!/ ----          ----          ---          ---'), ! M8
; P 0228 1      FMT_DCOMP_ERR2= ASCII
; 0229 1      ('!/!XL(X)      !XW(X)      !XW(X)      !XW(X)'),
; P 0230 1      FMT_EXT_PROC= ASCII
; 0231 1      ('!/          UBI IN EXTERNAL PROCESSOR MODE'),
; P 0232 1      FMT_INTERLOCK= ASCII
; 0233 1      ('!/CPU OR UBI FAILED TO LOCK ON DATIP FROM UET'), ! A10
; P 0234 1      FMT_INIT_ACLO= ASCII
; P 0235 1      ('!/CSR1 INIT DID NOT CLEAR ACLOW INTERRUPT ENABLE',
; P 0236 1      '!/EXPECTED: !XW(X)',
; 0237 1      '!/RECEIVED: !XW(X)'),
; P 0238 1      FMT_MAP_ADDR= ASCII
; 0239 1      ('!/MAP ADDRESS BUS BIT !UB STUCK'),
; P 0240 1      FMT_MAP_BIT_SA0=ASCII
; 0241 1      ('MAP ENTRY BIT STUCK AT ZERO!/' ),      ! M13
; P 0242 1      FMT_MAP_BIT_SA1=ASCII      ! A13
; 0243 1      ('MAP ENTRY BIT STUCK AT ONE!/' ),      ! A13
; P 0244 1      FMT_MAP_GEN= ASCII
; 0245 1      ('MAP NUMBER !XW(X) MAP ADDRESS: !XL(X)!/' ), ! A13
; P 0246 1      FMT_MAP_CS= ASCII
; 0247 1      ('!/MAP CHIP SELECT LINE STUCK'),
; P 0248 1      FMT_MAP_DB_SA0= ASCII      ! M8
; 0249 1      ('MAP DATA BUS BIT STUCK AT ZERO!/' ),      ! M8 M13
; P 0250 1      FMT_MAP_DB_SA1= ASCII      ! A8
; 0251 1      ('MAP DATA BUS BIT STUCK AT ONE!/' ),      ! A8 M13
; P 0252 1      FMT_MAP_FAIL= ASCII
; 0253 1      ('MAP FAILURE!/' ),      ! M13
; P 0254 1      FMT_MAP_INV= ASCII
; 0255 1      ('!/NO NXM FOR INVALID MAP'),
; P 0256 1      FMT_MCHK= ASCII
; P 0257 1      ('!/MACHINE CHECK OCCURRED WHILE READING OR WRITING',
; 0258 1      '    A REGISTER'),
; P 0259 1      FMT_NO_INT= ASCII
; 0260 1      ('!/NO INTERRUPT AT BR!UB WITH IPL AT !UB'),
; P 0261 1      FMT_NO_ACINT= ASCII
; P 0262 1      ('!/NO ACLOW ASCERTION INTERRUPT AT BR!UB WITH',
; 0263 1      '    IPL AT !UB'),
; P 0264 1      FMT_NO_ACINT1= ASCII
; P 0265 1      ('!/NO ACLOW DEASCERTION INTERRUPT AT BR!UB WITH',
; 0266 1      '    IPL AT !UB'),
; P 0267 1      FMT_NO_MCHK= ASCII      ! A8
; 0268 1      ('!/NO MACHINE CHECK WHEN UET ASCERTING PB'), ! A8
; P 0269 1      FMT_NO_UBE= ASCII
; 0270 1      ('!/NO UBES SELECTED. SKIPPING TEST'),      ! A8
; P 0271 1      FMT_NO_UBI= ASCII

```

```

; 0272 1      ('!/NO DW750s SELECTED. ABORTING PROGRAM'),
; P 0273 1      FMT_NOTHING= ASCII
; 0274 1      (' ZERO'),
; P 0275 1      FMT_UA1= ASCII
; P 0276 1      ('!/UNIBUS ADDRESS BIT ONE STUCK',
; P 0277 1      '!/EXPECTED: !XL(X)',
; 0278 1      '!/RECEIVED: !XL(X)'),
; P 0279 1      FMT_UB_CMI= ASCII
; P 0280 1      ('!/UNIBUS TO CMI MAPPING ERROR',
; P 0281 1      '!/EXPECTED: !XL(X)',
; 0282 1      '!/RECEIVED: !XL(X)'),
; P 0283 1      FMT_UBE_DUMP= ASCII
; P 0284 1      ('!/UBE(!UB) REGISTER DUMP:',
; P 0285 1      '!/DATA BUFFER: !XW(X)',
; P 0286 1      '!/CYCLE COUNT: !XW(X)',
; P 0287 1      '!/BUS ADDRESS: !XW(X)',
; P 0288 1      '!/CNTRL REG 1: !XW(X)',
; 0289 1      '!/CNTRL REG 2: !XW(X)'),
; P 0290 1      FMT_UBI_DUMP= ASCII
; P 0291 1      ('!/UBI CSR DUMP:',
; P 0292 1      '!/CSR 0: !XL(X)',
; P 0293 1      '!/CSR 1: !XL(X)',
; P 0294 1      '!/CSR 2: !XL(X)',
; 0295 1      '!/CSR 3: !XL(X)'),
; P 0296 1      FMT_UET_ACLO1= ASCII
; P 0297 1      ('!/UET CSR1 ACLOW1 NOT IN CORRECT STATE',
; P 0298 1      '!/EXPECTED: !XW(X)',
; 0299 1      '!/RECEIVED: !XW(X)'),
; P 0300 1      FMT_UET_INTD= ASCII
; P 0301 1      ('!/UET CSR1 INTERRUPT DONE BIT INCORRECT',
; P 0302 1      '!/EXPECTED: !XW(X)',
; 0303 1      '!/RECEIVED: !XW(X)'),
; P 0304 1      FMT_UET_INTD2= ASCII
; P 0305 1      ('!/UET CSR2 INTERRUPT DONE BIT INCORRECT',
; P 0306 1      '!/EXPECTED: !XW(X)',
; 0307 1      '!/RECEIVED: !XW(X)'),
; P 0308 1      FMT_UET_PB= ASCII
; 0309 1      ('!/PB BIT IN UET DID NOT CLEAR WITH UET INIT.'), ! A8
; P 0310 1      FMT_UET_STAT= ASCII
; P 0311 1      ('!/UET STATUS ERROR',
; P 0312 1      '!/EXPECTED: !XW(X)',
; 0313 1      '!/RECEIVED: !XW(X)'),
; P 0314 1      FMT_UB_TO= ASCII
; 0315 1      ('!/CPU TO UNIBUS TIMEOUT AT ADDRESS: !XL(X)'), ! A2
; P 0316 1      FMT_UNX_INT= ASCII
; 0317 1      ('!/UNEXPECTED UET INTERRUPT OCCURRED AT PC: !XL'),
; P 0318 1      FMT_UNX_INT1= ASCII
; 0319 1      ('!/UNEXPECTED INTERRUPT AT BR!UB WITH IPL AT !UB'),
; P 0320 1      FMT_UNX_ACINT= ASCII
; P 0321 1      ('!/UNEXPECTED ACLOW INTERRUPT AT BR!UB WITH',
; 0322 1      'IPL AT !UB'),
; P 0323 1      FMT_UNX_ACINT1= ASCII
; P 0324 1      ('!/UNEXPECTED ACLOW INTERRUPT WITH INTERRUPT ENABLE',
; 0325 1      'CLEAR'),
; P 0326 1      FMT_UNX_ACINT2= ASCII

```

```

; 0327 1      ('! /IO RESET DID NOT CLEAR ACLOW1 IN CSR1'),
; P 0328 1      FMT_VECTOR=  ASCII  ('! /INCORRECT INTERRUPT VECTOR',      ! A8
; P 0329 1      ('! /EXPECTED: !XW(X)',      ! A8
; P 0330 1      ('! /RECEIVED: !XW(X)'),      ! A8
; 0331 1
; P 0332 1      REC_MSG=  ASCII
; 0333 1      (' RECEIVED: '),
; P 0334 1      XOR_MSG=  ASCII
; 0335 1      (' XOR: '),
; P 0336 1      EXP_REC_XOR= ASCII
; P 0337 1      (' EXPECTED: !XL(X)!/',      ! A13
; P 0338 1      (' RECEIVED: !XL(X)!/',      ! A13
; 0339 1      (' XOR: !XL(X)!/'),      ! A13
; 0340 1
; 0341 1      !
; 0342 1      ! Chip callout lists
; 0343 1      !
; 0344 1
; 0345 1      UBIMOD=  ASCII ('FAILING MODULE: UBI'),      ! M7
; 0346 1      UBIMOD_BDP= ASCII ('FAILING ICs: UDP0-3'),      ! M7
; 0347 1      UBIMOD_BYTE_OFF= ASCII ('FAILING ICs: '),      ! M7
; 0348 1      UBIMOD_CMI= ASCII ('FAILING ICs: UDP0-3'),      ! M7
; 0349 1      UBIMOD_CPU_RW= ASCII ('FAILING ICs: UDP0-3'),      ! M7
; 0350 1      UBIMOD_CSR= ASCII ('FAILING ICs: UDP0-3, UCN'),      ! M7
; 0351 1      UBIMOD_DDP= ASCII ('FAILING ICs: UDP0-3'),      ! M7
; 0352 1      UBIMOD_DP_SEL= ASCII ('FAILING ICs: UDP0-3'),      ! M7
; 0353 1      UBIMOD_MAP= ASCII ('FAILING ICs: MAP RAMs'),      ! M7
; 0354 1      UBIMOD_MAP_ADDR= ASCII ('FAILING ICs: UDP0-3'),      ! M7
; 0355 1      UBIMOD_MAP_CS= ASCII ('FAILING ICs: UDP0-3'),      ! M7
; 0356 1      UBIMOD_MAP_CTRL= ASCII ('FAILING ICs: UDP0-3'),      ! M7
; 0357 1      UBIMOD-UA1= ASCII ('FAILING ICs: '),      ! M7
; 0358 1      UBIMOD-UB_CMI= ASCII ('FAILING ICs: '),      ! M7
; 0359 1      UETMOD=  ASCII ('FAILING MODULE: UET'),      ! A8
; 0360 1
; 0361 1      !
; 0362 1      ! Bit mnemonic strings
; 0363 1      !
; P 0364 1      CSR_MNEM=  ASCII
; 0365 1      ('ERR,NXM,UCE,UNDEFINED=28^X,PURGE'),
; P 0366 1      UBECR1_MNEM= ASCII
; 0367 1      ('ERR,DATOB_ON_DATIP,INH_DATIP,ROL,CC+DATA,FUNB,',
; 0368 1      'FUNA,C1,C0,RDY,INT_ODNE,NPR,BR7,BR6,BR5,BR4,GO'),
; P 0369 1      UBECR2_MNEM= ASCII
; 0370 1      ('BIT15,TM_DLY,CCOVF,ENB_PB,INTR_SSYN_ERR,',
; 0371 1      'NOGNT+NOTONEGNT,WRG_A_LINES,NO_SSYN_ERR,',
; P 0372 1      'NO_NO_SACK_ERR,MAX_LATE_ERR,WNG_GNT_BACK,',
; 0373 1      'PWR_DWN_SEQ,INH_SACK,INH_INC_ADDR_CC,A17,A16');
; 0374 1
; 0375 1      ! OWN STORAGE:
; 0376 1      !
; 0377 1
; 0378 1      STRUCTURE
; 0379 1      ERROR_BUFFER[1, 0; N, BS, UNIT=4] =
; 0380 1      [N*BS*UNIT]
; 0381 1      (ERROR_BUFFER+(0+I*BS)*UNIT);

```



```

: 0382 1
: 0383 1
: 0384 1
: 0385 1
: 0386 1
: 0387 1
: 0388 1
: 0389 1
: 0390 1
: 0391 1
: 0392 1
: 0393 1
: 0394 1
: 0395 1
: 0396 1
: 0397 1
: 0398 1
: 0399 1
: 0400 1
: 0401 1
: 0402 1
: 0403 1
: 0404 1
: 0405 1
: 0406 1
: 0407 1
: 0408 1
: 0409 1
: 0410 1
: 0411 1
: 0412 1
: 0413 1
: 0414 1
: 0415 1
: 0416 1
: 0417 1
: 0418 1
: 0419 1
: 0420 1
: 0421 1
: 0422 1
: 0423 1
: 0424 1
: 0425 1
: 0426 1
: 0427 1
: 0428 1
: 0429 1
: 0430 1
: 0431 1
: 0432 1
: 0433 1
: 0434 1
: 0435 1
: 0436 1

GLOBAL

CHAN_STAT: VECTOR[2, LONG],
CODEWORDS: VECTOR[9, WORD],
CODEVECTORS: VECTOR[4, WORD],

UBI_UNDER_TEST,
UBI_VECTOR: WORD,
UBI_LN: VECTOR[2],
UNIBUS_SPACE,
DECODED_WORDS: VECTOR[9],
EVENT_FLAG: BITVECTOR[4],
EXTERNAL_FLAG: BYTE,
FREE_MAP: VECTOR[4, WORD],
NUM_UBI: BYTE,
UBE_SEL: BYTE,
DEVICES: BYTE,
LN: BYTE,
INT_VECTOR,
INTERRUPT_EXP: BYTE,
INTERRUPT_OCC: BYTE,
LUN: BYTE,
NOISY_WORDS: VECTOR[9, WORD],
NUM_DATA_ERR: VECTOR[4],
NUM_MAPS: WORD,
NUM_UBE,
PTBASE: REF $DS_HPO_DECL(),
UB_MAP: BITVECTOR[31],
UBE_BASE_ADDR: VECTOR[4],
UBE_BUF_ADDR: VECTOR[4],

UBE_MODE: VECTOR[4, BYTE],

UBE_MAP_NO: VECTOR[4, WORD],

UBE_DP_NO: VECTOR[4, BYTE],
UBE_BUF_SIZE,
UBE_PAGE_CNT,

UBE_ERR_BUF:
    ERROR_BUFFER
    [4, 64],
UBE_INIT_DATA: VECTOR[4, WORD],
UBE_EXP_DATA: VECTOR[4, WORD],

UBE_INT_OCC: VECTOR[4, BYTE],
UBE_SERVICE: VECTOR[4, LONG],
UBE_XFER_ERR: BITVECTOR[4],
UBE_XFER_SIZE: VECTOR[4, LONG],

UBE_STAT_ERR: BITVECTOR[4],
UBE_VECTOR: VECTOR[4, WORD],
UBE_DRIVE: VECTOR[4, BYTE],
UBI_LINK: REF $DS_HPO_DECL(),
VALID_MAPS: BITVECTOR[496];

! Receives Channel Service Status
! Receives Reed-Muller encoded words
! Contains vectors forming basis of
! Reed-Muller (16,5) coding space
! Receives phys addr of UBI under test
! Base offset of UBI vector space

! Receives phys addr of Unibus Space
! Contains the nine decoded words
! Indicates UBE interrupt completion
! External Processor timeout flag.
! Contains no. of first available maps
! Contains no. of dw's

! Receives the UET interrupt vector
! Interrupt expected flag
! Interrupt occurred flag for UET
! Logical unit number
! Table of possibly corrupted codewords
! Number of bad words in a UBE xfer
! Contains total no. of useable maps
! Number of UBES on system
! Receives base address of P-Table
! Bit map of Unibus memory
! Contains base address of UBE[N]
! Contains address of I/O buffer
! associated with UBE[N]
! Contains flags to indicate UBE[N] is
! operating in byte offset mode
! Contains the base map number for the
! UBE transfer
! Data path number assigned to that UBE
! Contains size in pages of IO buffers
! Contains number of pages taken from
! Diagnostic Supervisor
! A list of pointers to actual (bad)
! data in the I/O buffer associated
! with UBE[N]
! Contains initial data in buffer A11
! Contains data expected in buffer
! associated with UBE[N]
! Interrupt occurred flags for UBE's ! A8
! Gets address of UBE Service Routines A8
! Set if error detected in transfer
! Contains number of words
! used by UBE[N] in transfer
! Indicates UBE[N] has status error
! UBE Interrupt Vectors
! UBE Drive Numbers

! Represents useable maps

```

```
; 0437 1
; 0438 1      !+
; 0439 1      ! This establishes a page-aligned temporary buffer:
; 0440 1      !--
; 0441 1
; 0442 1      PSECT
; 0443 1          GLOBAL = SCRATCHPAD(ALIGN(9));
; 0444 1      GLOBAL
; 0445 1          SCRATCH: VECTOR[128] ALIGN(9); ! A scratchpad for temporary storage
; 0446 1
; 0447 1      !
; 0448 1      ! EXTERNAL REFERENCES:
; 0449 1      !
; 0450 1
; 0451 1      EXTERNAL ROUTINE
; 0452 1          UBE0_ISR: ADDRESSING_MODE(LONG_RELATIVE),
; 0453 1          UBE1_ISR: ADDRESSING_MODE(LONG_RELATIVE),
; 0454 1          UBE2_ISR: ADDRESSING_MODE(LONG_RELATIVE),
; 0455 1          UBE3_ISR: ADDRESSING_MODE(LONG_RELATIVE);
```

INITIALIZE

```

: 0456 2 $SDS_BGNINIT
: 0457 2 !**
: 0458 2 ! FUNCTIONAL DESCRIPTION:
: 0459 2 !
: 0460 2 !     The initialization routine is the first module called by the
: 0461 2 !     Diagnostic Supervisor upon startup of the diagnostic program. It is
: 0462 2 !     called again at the end of the test sequence to determine if end of
: 0463 2 !     pass has been reached. The routine captures all of the selected UBE
: 0464 2 !     vectors and determines which UBIs have been selected for test by the
: 0465 2 !     user. It also determines which UBEs are to be used.
: 0466 2 !
: 0467 2 ! FORMAL PARAMETERS: NONE
: 0468 2 !
: 0469 2 ! IMPLICIT INPUTS: NONE
: 0470 2 !
: 0471 2 ! IMPLICIT OUTPUTS: NONE
: 0472 2 !
: 0473 2 ! COMPLETION CODES: NONE
: 0474 2 !
: 0475 2 ! SIDE EFFECTS:
: 0476 2 !
: 0477 2 !     The vectors of all UBEs selected for use are captured.
: 0478 2 ! --
: 0479 2
: 0480 2 LOCAL
: 0481 2     POINTER_1,
: 0482 2     POINTER_2,
: 0483 2     UBE,
: 0484 2     UBE_ADDR:VECTOR[4],
: 0485 2     TEMP;
: 0486 2
: 0487 2 LITERAL
: 0488 2     IO_RESET=  %X'37';
: 0489 2
: 0490 2 BUILTIN
: 0491 2     MTPR;
: 0492 2
: 0493 2 TEMP = -1;
: 0494 2 MTPR(TEMP,IO_RESET);
: 0495 2
: 0496 2 CODECOMMENT
: 0497 2 ''
: 0498 2 'One p-table is associated with each logical number (LN). The user',
: 0499 2 'attach and select UBIs and UBEs in any order. Therefore, p-tables for UBIs',
: 0500 2 'and UBEs may be in any order. Because of this, it is necessary to',
: 0501 2 'determine the device type for each p-table to ascertain what action is to',
: 0502 2 'be taken with a particular p-table.',
: 0503 2 ''
: 0504 2 'All existing p-tables are scanned, looking for UBIs (DW750s). The first',
: 0505 2 'UBI found is assigned as UBI logical number 0. After all UBIs are found',
: 0506 2 'and assigned logical numbers, the p-tables are again searched for UBEs.',
: 0507 2 'Each UBE which is associated with the current UBI under test is assigned',
: 0508 2 'with a logical number. The UBE address is stored in the vector',
: 0509 2 'UBE_BASE_ADDR and its vector is captured.',
: 0510 2 ''

```



```

; 0511 2 ![[1.4] Reminder: If two UBI's are being tested then one Supervisor Pass is equal
; 0512 2 ![[1.4] to execution of UBI0 INIT then UBI0 TEST then UBI1 INIT then UBI1 TEST.
; 0513 2 ![[1.4] The ONLY time DSA$V_PASS0 is set is during UBI0 INIT on the first
; 0514 2 ![[1.4] Supervisor Pass.
; 0515 2
; 0516 3 BEGIN
; 0517 3 IF .DSA$V_PASS0
; 0518 3 THEN
; 0519 4 BEGIN
; 0520 4 LUN = - 1;
; 0521 4 LN = 0;
; 0522 4 UBE = 'UBE';
; 0523 4 UBE<0,8> = 3;
; 0524 4 UBE_SEL = 0;
; 0525 4 NUM_UBI = 0;
; 0526 4 DEVICES = 0;
; 0527 4 WHILE $DS_GPHARD(UNIT=.LN,RETADR=PTBASE) DO ! Get all the logical units.
; 0528 5 BEGIN
; 0529 5 IF .(PTBASE[HP$T_TYPE])<0,32> EQL .UBE ! If UBE found then set flag
; 0530 5 THEN
; 0531 5 UBE_SEL = 1
; 0532 5 ELSE
; 0533 6 BEGIN
; 0534 6 UBI_LN[.NUM_UBI] = .LN;
; 0535 6 NUM_UBI = .NUM_UBI + 1;
; 0536 5 END;
; 0537 5 LN = .LN + 1;
; 0538 4 END;
; 0539 4
; 0540 4 IF .NUM_UBI EQL 0
; 0541 4 THEN
; 0542 5 BEGIN
; 0543 5 $DS_PRINTX(FMT_NO_UBI);
; 0544 5 $DS_ABORT(ARG = PROGRAM);
; 0545 4 END;
; 0546 4
; 0547 4 DEVICES = .LN;
; 0548 4 LN = 0;
; 0549 4 NUM_UBE = 0;
; 0550 4 UBE_SERVICE[0] = UBE0_ISR;
; 0551 4 UBE_SERVICE[1] = UBE1_ISR;
; 0552 4 UBE_SERVICE[2] = UBE2_ISR;
; 0553 4 UBE_SERVICE[3] = UBE3_ISR;
; 0554 4 EXTERNAL_FLAG = 0;
; 0555 3 END;
; 0556 3
; 0557 3 ! [[1.4] Code above executed only for First Init of First Pass.
; 0558 3
; 0559 3 LUN = .LUN + 1;
; 0560 3 IF .LUN GEQ .NUM_UBI
; 0561 3 THEN
; 0562 4 BEGIN
; 0563 4 $DS_ENDPASS;
; 0564 4 LUN = 0;
; 0565 3 END;

```

![[1.4] DSA\$V_PASS0 is set before
![[1.4] and cleared immediately
![[1.4] after the Init code is
![[1.4] executed for the first time
![[1.4] in the first pass. It is
![[1.4] clear from then on.

! else increment number
! of UBI's.
! Setting up an array:
! UBI_LN[0] = first LN
! UBI_LN[1] = next LN

! If there are no UBI's
! then print error.

! Initializations.
! Interrupt service
! routines.

! Increment UBI logical unit
! number. If all UBI's tested
! then execute end of pass.

! [[1.4] \$DS_ENDPASS returns
! [[1.4] here. Clear LUN because
! [[1.4] you want to setup for

ZZ-ECCBA-1.4
UBI_HEADER
01-04

INITIALIZE
INITIALIZE

K 5

6-Sep-1985

Fiche 1 Frame K5

Sequence 62

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 13

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21

(4)

```
: 0566 3
: 0567 3
: 0568 3
: 0569 3
: 0570 3
: 0571 3
: 0572 3
: 0573 3
: 0574 3
: 0575 3
: 0576 3
: 0577 4
: 0578 4
: 0579 5
: 0580 5
: 0581 5
: 0582 5
: 0583 6
: 0584 6
: 0585 6
: 0586 7
: 0587 7
: 0588 7
: 0589 7
: P 0590 7
: 0591 7
: 0592 7
: 0593 6
: 0594 5
: 0595 4
: 0596 3
: 0597 2
: 0598 2
: 0599 1

$DS_GPHARD(UNIT = .UBI_LN[.LUN],RETADR = PTBASE);
    UBI_UNDER_TEST = .PTBASE[HP$A_DEVICE];
    UNIBUS_SPACE = .PTBASE[HP$A_DVA];
    UBI_LINK = .PTBASE;
    UBI_VECTOR = .PTBASE[HP$W_VECTOR];

IF .UBE_SEL
THEN
    BEGIN
        INCR LN FROM 0 TO .DEVICES - 1 DO
            BEGIN
                $DS_GPHARD(UNIT = .LN,RETADR = PTBASE);
                IF .(PTBASE[HP$T_TYPE])<0,32> EQL .UBE
                THEN
                    BEGIN
                        IF .PTBASE[HP$A_LINK] EQL .UBI_LINK
                        THEN
                            BEGIN
                                UBE_BASE_ADDR[.NUM_UBE] = .PTBASE[HP$A_DEVICE];
                                UBE_VECTOR[.NUM_UBE] = .PTBASE[HP$W_VECTOR] OR .UBI_VECTOR;
                                UBE_DRIVE[.NUM_UBE] = .PTBASE[HP$B_DRIVE];
                                $DS_SETVEC(VECTOR = .UBE_VECTOR[.NUM_UBE],
                                    SRVADR = .UBE_SERVICE[.NUM_UBE]);
                                NUM_UBE = .NUM_UBE + 1;
                            END;
                        END;
                    END;
                END;
            END;
        END;
    END;
$DS_ENDINIT ;
```

! [1.4] UBI0 before executing
! [1.4] tests on next pass.

! Setup for a UBI

! If flag was set above
! then there are UBE's.
! So setup for them.

.TITLE UBI_HEADER
.IDENT \01-04\
.PSECT SCRATCHPAD,NOEXE,9

00000 SCRATCH::

.BLKB 512
.PSECT \$HEADER,NOWRT,NOEXE,9

```
00000005 00000058 00000000 00000000
00000004 00000001 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000V 00000000V 00000000 00000000
00000000 00000000 00000000 00000000
00000000V 00000000 00000000 00000000

HEADER: .LONG 88, 5
        .ADDRESS P.AAH
        .LONG 1, 4
        .ADDRESS LASTAD, DISPATCH, AL_DEVTyp
        .LONG 10, 0
        .LONG 0[5]
        .ADDRESS INITIALIZE, CLEAN_UP, SUMMARY
        .LONG 0, 0
        .ADDRESS SECTION, L_TSTCNT
```

ZZ-ECCBA-1.4
UBI_HEADER
01-04

INITIALIZE
INITIALIZE

L 5

6-Sep-1985

Fiche 1 Frame L5

Sequence 63

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 14

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21

(4)

```
.PSECT ____LAST,2
00000 LASTAD: .BLKB 0
.PSECT DISPATCH_X,NOWRT,NOEXE,2
00000000# 00000 $: .LONG 0[6]
.PSECT DISPATCH,NOWRT,NOEXE,2
00000 DISPATCH:
.BLKB 0
.PSECT GLOBALS,NOEXE,2
00000 CHAN_STAT::
.BLKB 8
00008 CODEWORDS::
.BLKB 18
0001A .BLKB 2
0001C CODEVECTORS::
.BLKB 8
00024 UBI_UNDER_TEST::
.BLKB 4
00028 UBI_VECTOR::
.BLKB 2
0002A .BLKB 2
0002C UBI_LN:: .BLKB 8
00034 UNIBUS_SPACE::
.BLKB 4
00038 DECODED_WORDS::
.BLKB 36
0005C EVENT_FLAG::
.BLKB 1
0005D EXTERNAL_FLAG::
.BLKB 1
0005E .BLKB 2
00060 FREE_MAP::
.BLKB 8
00068 NUM_UBI::
.BLKB 1
00069 UBE_SEL::
.BLKB 1
0006A DEVICES::
.BLKB 1
0006B LN:: .BLKB 1
0006C INT_VECTOR::
.BLKB 4
00070 INTERRUPT_EXP::
.BLKB 1
00071 INTERRUPT_OCC::
.BLKB 1
00072 LUN:: .BLKB 1
00073 .BLKB 1
```


2
ZZ-ECCBA-1.4
UBI_HEADER
01-04

INITIALIZE
INITIALIZE

M 5

6-Sep-1985

Fiche 1

Frame M5

Sequence 64

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 15

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21

(4)

```
00074 NOISY_WORDS::
      .BLKB 18
00086      .BLKB 2
00088 NUM_DATA_ERR::
      .BLKB 16
00098 NUM_MAPS::
      .BLKB 2
0009A      .BLKB 2
0009C NUM_UBE::
      .BLKB 4
000A0 PTBASE::.BLKB 4
000A4 UB_MAP::.BLKB 4
000A8 UBE_BASE_ADDR::
      .BLKB 16
000B8 UBE_BUF_ADDR::
      .BLKB 16
000C8 UBE_MODE::
      .BLKB 4
000CC UBE_MAP_NO::
      .BLKB 8
000D4 UBE_DP_NO::
      .BLKB 4
000D8 UBE_BUF_SIZE::
      .BLKB 4
000DC UBE_PAGE_CNT::
      .BLKB 4
000E0 UBE_ERR_BUF::
      .BLKB 1024
004E0 UBE_INIT_DATA::
      .BLKB 8
004E8 UBE_EXP_DATA::
      .BLKB 8
004F0 UBE_INT_OCC::
      .BLKB 4
004F4 UBE_SERVICE::
      .BLKB 16
00504 UBE_XFER_ERR::
      .BLKB 1
00505      .BLKB 3
00508 UBE_XFER_SIZE::
      .BLKB 16
00518 UBE_STAT_ERR::
      .BLKB 1
00519      .BLKB 3
0051C UBE_VECTOR::
      .BLKB 8
00524 UBE_DRIVE::
      .BLKB 4
00528 UBI_LINK::
      .BLKB 4
0052C VALID_MAPS::
      .BLKB 62
      .PSECT $PLIT$,NOWRT,NOEXE,2
```

54	4C	55	41	46	45	44	07	00000	P.AAB:	.ASCII	<7>\DEFAULT\	:								
				4C	4C	41	03	00008	P.AAC:	.ASCII	<3>\ALL\	:								
				45	42	55	03	0000C	P.AAD:	.ASCII	<3>\UBE\	:								
								00010		.LONG	3	:								
00000000				00000000		00000000		00014	P.AAA:	.ADDRESS	P.AAB, P.AAC, P.AAD	:								
30		35	37	57	44	05	00	00020	P.AAF:	.ASCII	<5>\DW750\	:								
								00026		.BYTE	0	:								
								00027		.BLKB	1	:								
				45	42	55	03	00028	P.AAG:	.ASCII	<3>\UBE\	:								
							00	0002C		.BYTE	0	:								
								0002D		.BLKB	3	:								
								00030		.LONG	2	:								
				00000000		00000000		00034	P.AAE:	.ADDRESS	P.AAF, P.AAG	:								
34	2E	31	20	2E	56	45	52	2D	41	42	43	43	45	32	0003C	P.AAH:	.ASCII	\2ECCBA-REV. 1.4	VAX 11/750(UBI), DW750\	:
55	28	30	35	37	2F	31	31	20	58	41	56	20	20	20	0004B					:
					30	35	37	57	44	20	2C	29	49	42	0005A					:
			00	63	69	74	73	6F	6E	67	61	69	44	20	00064		.ASCII	\ Diagnostic\<0>		:
			20	3A	44	45	54	43	45	50	58	45	20	0B	00070	P.AAI:	.ASCII	<11>\ EXPECTED: \		:
20	20	29	58	28	4C	58	21	20	43	41	21	2F	21	13	0007C	P.AAJ:	.ASCII	<19>\!/\!AC !XL(X) ; !AC\		:
										43	41	21	20	3B	0008B					:
55	21	20	48	54	41	50	20	41	54	41	44	2F	21	3E	00090	P.AAK:	.ASCII	\>!/DATA PATH !UB DATI ERROR!/EXPECTED: !\		:
45	2F	21	52	4F	52	52	45	20	49	54	41	44	20	42	0009F					:
					21	20	3A	44	45	54	43	45	50	58	000AE					:
44	45	56	49	45	43	45	52	2F	21	29	58	28	57	58	000B8		.ASCII	\XW(X)!/RECEIVED: !XW(X)\		:
							29	58	28	57	58	21	20	3A	000C7					:
55	21	20	48	54	41	50	20	41	54	41	44	2F	21	3E	000CF	P.AAL:	.ASCII	\>!/DATA PATH !UB DATO ERROR!/EXPECTED: !\		:
45	2F	21	52	4F	52	52	45	20	4F	54	41	44	20	42	000DE					:
					21	20	3A	44	45	54	43	45	50	58	000ED					:
44	45	56	49	45	43	45	52	2F	21	29	58	28	4C	58	000F7		.ASCII	\XL(X)!/RECEIVED: !XL(X)\		:
							29	58	28	4C	58	21	20	3A	00106					:
55	21	20	48	54	41	50	20	41	54	41	44	2F	21	3F	0010E	P.AAM:	.ASCII	\?!/DATA PATH !UB DATOB ERROR!/EXPECTED: \		:
2F	21	52	4F	52	52	45	20	42	4F	54	41	44	20	42	0011D					:
					20	3A	44	45	54	43	45	50	58	45	0012C					:
45	56	49	45	43	45	52	2F	21	29	58	28	4C	58	21	00136		.ASCII	\!XL(X)!/RECEIVED: !XL(X)\		:
						29	58	28	4C	58	21	20	3A	44	00145					:
55	42	49	4E	55	20	4F	54	20	49	4D	43	2F	21	41	0014E	P.AAN:	.ASCII	\A!/CMI TO UNIBUS MAPPING ERROR!/EXPECTED\		:
52	4F	52	52	45	20	47	4E	49	50	50	41	4D	20	53	0015D					:
					44	45	54	43	45	50	58	45	2F	21	0016C					:
49	45	43	45	52	2F	21	29	58	28	4C	58	21	20	3A	00176		.ASCII	\: !XL(X)!/RECEIVED: !XL(X)\		:
				29	58	28	4C	58	21	20	3A	44	45	56	00185					:
55	42	49	4E	55	20	4F	54	20	49	4D	43	2F	21	41	00190	P.AAO:	.ASCII	\A!/CMI TO UNIBUS MAPPING ERROR!/EXPECTED\		:
52	4F	52	52	45	20	47	4E	49	50	50	41	4D	20	53	0019F					:
					44	45	54	43	45	50	58	45	2F	21	001AE					:
49	45	43	45	52	2F	21	29	58	28	4C	58	21	20	3A	001B8		.ASCII	\: !XL(X)!/RECEIVED: !XL(X)\		:
				29	58	28	4C	58	21	20	3A	44	45	56	001C7					:
49	52	57	2F	44	41	45	52	20	55	50	43	2F	21	3A	001D2	P.AAP:	.ASCII	\:!/CPU READ/WRITE ERROR!/EXPECTED: !XW(X)\		:
43	45	50	58	45	2F	21	52	4F	52	52	45	20	45	54	001E1					:
					58	28	57	58	21	20	3A	44	45	54	001F0					:
58	21	20	3A	44	45	56	49	45	43	45	52	2F	21	29	001FA		.ASCII	\)!/RECEIVED: !XW(X)\		:
										29	58	28	57	00209						:
52	4F	52	52	45	20	42	55	21	52	53	43	2F	21	0E	0020D	P.AAQ:	.ASCII	<14>\!/CSR!UB ERROR\		:
21	20	3D	20	29	42	55	21	52	53	43	28	2F	21	13	0021C	P.AAR:	.ASCII	<19>\!/((CSR!UB) = !XL(X)\		:
										29	58	28	4C	58	0022B					:
52	52	45	20	49	54	41	44	20	50	44	44	2F	21	34	00230	P.AAS:	.ASCII	\4!/DDP DATI ERROR!/EXPECTED: !XW(X)!/REC\		:
21	20	3A	44	45	54	43	45	50	58	45	2F	21	52	4F	0023F					:

		29	58	28	43	45	52	2F	21	29	58	28	57	58	0024E				
52	52	45	20	4F	54	41	44	20	3A	44	45	56	49	45	00258	P.AAT:	.ASCII	\EIVED: !XW(X)\	
53	53	45	52	44	44	41	20	49	4D	43	2F	21	52	4F	00265		.ASCII	\O!/DDP DATO ERROR!/CMI ADDRESS IN ERROR:\	
					3A	52	4F	52	52	45	20	4E	49	20	00274				
3A	44	45	54	43	45	50	58	45	2F	21	4C	58	21	20	00283				
56	49	45	43	45	52	2F	21	29	58	28	57	58	21	20	0028D		.ASCII	\ !XL!/EXPECTED: !XW(X)!/RECEIVED: !XW(X)\	
					29	58	28	57	58	21	20	3A	44	45	0029C				
52	45	20	42	4F	54	41	44	20	50	44	44	2F	21	66	002AB	P.AAU:	.ASCII	\f!/DDP DATOB ERROR!/CMI ADDRESS IN ERROR\	
53	45	52	44	44	41	20	49	4D	43	2F	21	52	4F	52	002B5				
					52	4F	52	52	45	20	4E	49	20	53	002C4				
20	45	54	59	42	2F	21	29	58	28	4C	58	21	20	3A	002D3				
45	2F	21	42	55	21	20	3A	4E	45	54	54	49	52	57	002DD		.ASCII	\: !XL(X)!/BYTE WRITTEN: !UB!/EXPECTED: !\	
					21	20	3A	44	45	54	43	45	50	58	002EC				
44	45	56	49	45	43	45	52	2F	21	29	58	28	57	58	002FB				
							29	58	28	57	58	21	20	3A	00305		.ASCII	\XW(X)!/RECEIVED: !XW(X)\	
45	53	20	48	54	41	50	20	41	54	41	44	2F	21	3C	00314	P.AAV:	.ASCII	\<!/DATA PATH SELECT ERROR!/EXPECTED: !XL\	
50	58	45	2F	21	52	4F	52	52	45	20	54	43	45	4C	0031C				
					4C	58	21	20	3A	44	45	54	43	45	0032B				
20	3A	44	45	56	45	49	43	45	52	2F	21	29	58	28	0033A				
									29	58	28	4C	58	21	00344		.ASCII	\(X)!/RECIEVED: !XL(X)\	
45	52	41	50	4D	4F	43	20	41	54	41	44	2F	21	B5	00353	P.AAW:	.ASCII	<181>\!/DATA COMPARE ERROR!/XFER SIZE: !\	
49	53	20	52	45	46	58	2F	21	52	4F	52	52	45	20	00359				
									21	20	3A	45	5A		00368				
54	53	20	20	53	45	54	59	42	20	29	58	28	4C	58	00377				
3A	53	53	45	52	44	44	41	20	47	4E	49	54	52	41	0037C		.ASCII	\XL(X) BYTES STARTING ADDRESS: !XL(X) \	
					20	20	20	29	58	28	4C	58	21	20	0038B				
44	41	42	20	53	44	52	4F	57	20	4C	41	54	4F	54	0039A				
23	20	45	42	55	2F	21	29	58	28	4C	58	21	20	3A	003A4		.ASCII	\TOTAL WORDS BAD: !XL(X)!/UBE #: !UB DAY	
					41	44	20	20	20	42	55	21	20	3A	003B3				
20	42	55	21	20	3A	23	20	48	54	41	50	20	41	54	003C2				

5
)

22-ECCBA-1.4
UBI HEADER
01-04

INITIALIZE
INITIALIZE

C 6

6-Sep-1985

Fiche 1 Frame C6

Sequence 67

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 18

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21

(4)

45	50	58	45	2F	21	45	4C	42	41	4E	45	20	54	50	00509	.ASCII	\PT ENABLE!/EXPECTED: !XW(X)!/RECEIVED: !\	:	
52	2F	21	29	58	28	57	58	21	20	3A	44	45	54	43	00518			:	
					21	20	3A	44	45	56	49	45	43	45	00527			:	
										29	58	28	57	58	00531			:	
20	53	53	45	52	44	44	41	20	50	41	4D	2F	21	1F	00536	P.ABC:	.ASCII	\XW(X)\	:
55	54	53	20	42	55	21	20	54	49	42	20	53	55	42	00545			<31>\!/MAP ADDRESS BUS BIT !UB STUCK\	:
													4B	43	00554				:
20	54	49	42	20	59	52	54	4E	45	20	50	41	4D	1D	00556	P.ABD:	.ASCII	<29>\MAP ENTRY BIT STUCK AT ZERO!\	:
2F	21	4F	52	45	5A	20	54	41	20	4B	43	55	54	53	00565				:
20	54	49	42	20	59	52	54	4E	45	20	50	41	4D	1C	00574	P.ABE:	.ASCII	<28>\MAP ENTRY BIT STUCK AT ONE!\	:
	2F	21	45	4E	4F	20	54	41	20	4B	43	55	54	53	00583				:
57	58	21	20	52	45	42	4D	55	4E	20	50	41	4D	28	00591	P.ABF:	.ASCII	\(MAP NUMBER !XW(X) MAP ADDRESS: !XL(X)!\	:
53	45	52	44	44	41	20	50	41	4D	20	20	29	58	28	005A0				:
					21	29	58	28	4C	58	21	20	3A	53	005AF				:
													2F	005B9			\/\		:
4C	45	53	20	50	49	48	43	20	50	41	4D	2F	21	1C	005BA	P.ABG:	.ASCII	<28>\!/MAP CHIP SELECT LINE STUCK\	:
	4B	43	55	54	53	20	45	4E	49	4C	20	54	43	45	005C9				:
42	20	53	55	42	20	41	54	41	44	20	50	41	4D	20	005D7	P.ABH:	.ASCII	\ MAP DATA BUS BIT STUCK AT ZERO!\	:
52	45	5A	20	54	41	20	4B	43	55	54	53	20	54	49	005E6				:
												2F	21	4F	005F5				:
42	20	53	55	42	20	41	54	41	44	20	50	41	4D	1F	005F8	P.ABI:	.ASCII	<31>\MAP DATA BUS BIT STUCK AT ONE!\	:
45	4E	4F	20	54	41	20	4B	43	55	54	53	20	54	49	00607				:
													2F	21	00616				:
	2F	21	45	52	55	4C	49	41	46	20	50	41	4D	0D	00618	P.ABJ:	.ASCII	<13>\MAP FAILURE!\	:
49	20	52	4F	46	20	4D	58	4E	20	4F	4E	2F	21	18	00626	P.ABK:	.ASCII	<24>\!/NO NXM FOR INVALID MAP\	:
					50	41	4D	20	44	49	4C	41	56	4E	00635				:
43	45	48	43	20	45	4E	49	48	43	41	4D	2F	21	3C	0063F	P.ABL:	.ASCII	\<!/MACHINE CHECK OCCURRED WHILE READING \	:
4C	49	48	57	20	44	45	52	52	55	43	43	4F	20	4B	0064E				:
					20	47	4E	49	44	41	45	52	20	45	0065D				:
45	52	20	41	20	47	4E	49	54	49	52	57	20	52	4F	00667			\OR WRITING A REGISTER\	:
										52	45	54	53	49	00676				:
54	50	55	52	52	45	54	4E	49	20	4F	4E	2F	21	27	0067C	P.ABM:	.ASCII	\!/NO INTERRUPT AT BR!UB WITH IPL AT !UB\	:
20	48	54	49	57	20	42	55	21	52	42	20	54	41	20	0068B				:
					42	55	21	20	54	41	20	4C	50	49	0069A				:
43	53	41	20	57	4F	4C	43	41	20	4F	4E	2F	21	37	006A4	P.ABN:	.ASCII	\7!/NO ACLOW ASCERTION INTERRUPT AT BR!UB\	:
50	55	52	52	45	54	4E	49	20	4E	4F	49	54	52	45	006B3				:
					42	55	21	52	42	20	54	41	20	54	006C2				:
55	21	20	54	41	20	4C	50	49	20	48	54	49	57	20	006CC			\ WITH IPL AT !UB\	:
													42	006DB					:
41	45	44	20	57	4F	4C	43	41	20	4F	4E	2F	21	39	006DC	P.ABO:	.ASCII	\9!/NO ACLOW DEASCERTION INTERRUPT AT BR!\	:
52	52	45	54	4E	49	20	4E	4F	49	54	52	45	43	53	006EB				:
					21	52	42	20	54	41	20	54	50	55	006FA				:
20	54	41	20	4C	50	49	20	48	54	49	57	20	42	55	00704			\UB WITH IPL AT !UB\	:
												42	55	21	00713				:
43	20	45	4E	49	48	43	41	4D	20	4F	4E	2F	21	28	00716	P.ABP:	.ASCII	\(!/NO MACHINE CHECK WHEN UET ASCERTING P\	:
41	20	54	45	55	20	4E	45	48	57	20	4B	43	45	48	00725				:
					50	20	47	4E	49	54	52	45	43	53	00734				:
														42	0073E			\B\	:
45	4C	45	53	20	53	45	42	55	20	4F	4E	2F	21	21	0073F	P.ABQ:	.ASCII	\!/NO UBES SELECTED. SKIPPING TEST\	:
20	47	4E	49	50	50	49	4B	53	20	2E	44	45	54	43	0074E				:
											54	53	45	54	0075D				:
45	53	20	73	30	35	37	57	44	20	4F	4E	2F	21	26	00761	P.ABR:	.ASCII	\&!/NO DW750s SELECTED. ABORTING PROGRAM\	:
4E	49	54	52	4F	42	41	20	2E	44	45	54	43	45	4C	00770				:
						4D	41	52	47	4F	52	50	20	47	0077F				:
						4F	52	45	5A	20	20	20	20	08	00788	P.ABS:	.ASCII	<8>\ ZERO\	:

ZZ-ECCBA-1.4
UBI HEADER
01-04INITIALIZE
INITIALIZE

D 6

6-Sep-1985

Fiche 1

Frame D6

Sequence 68

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 19

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21

(4)

45	52	44	44	41	20	53	55	42	49	4E	55	2F	21	42	00791	P.ABT:	.ASCII	\B!/UNIBUS ADDRESS BIT ONE STUCK!/EXPECTE\	:	
43	55	54	53	20	45	4E	4F	20	54	49	42	20	53	53	007A0				:	
					45	54	43	45	50	58	45	2F	21	4B	007AF				:	
45	43	45	52	2F	21	29	58	28	4C	58	21	20	3A	44	007B9		.ASCII	\D: !XL(X)!/RECEIVED: !XL(X)\	:	
			29	58	28	4C	58	21	20	3A	44	45	56	49	007C8				:	
4D	43	20	4F	54	20	53	55	42	49	4E	55	2F	21	41	007D4	P.ABU:	.ASCII	\A!/UNIBUS TO CMI MAPPING ERROR!/EXPECTED\	:	
52	4F	52	52	45	20	47	4E	49	50	50	41	4D	20	49	007E3				:	
					44	45	54	43	45	50	58	45	2F	21	007F2				:	
49	45	43	45	52	2F	21	29	58	28	4C	58	21	20	3A	007FC		.ASCII	\: !XL(X)!/RECEIVED: !XL(X)\	:	
				29	58	28	4C	58	21	20	3A	44	45	56	0080B				:	
47	45	52	20	5D	42	55	21	5B	45	42	55	2F	21	82	00816	P.ABV:	.ASCII	<130>\!/UBE[!UB] REGISTER DUMP:!/DATA BU\	:	
41	44	2F	21	3A	50	4D	55	44	20	52	45	54	53	49	00825				:	
										55	42	20	41	54	00834				:	
43	2F	21	29	58	28	57	58	21	20	3A	52	45	46	46	00839		.ASCII	\FFER: !XW(X)!/CYCLE COUNT: !XW(X)!/BUS A\	:	
57	58	21	20	3A	54	4E	55	4F	43	20	45	4C	43	59	00848				:	
					41	20	53	55	42	2F	21	29	58	28	00857				:	
21	29	58	28	57	58	21	20	3A	53	53	45	52	44	44	00861		.ASCII	\DDRESS: !XW(X)!/CNTRL REG 1: !XW(X)!/CNT\	:	
21	20	3A	31	20	47	45	52	20	4C	52	54	4E	43	2F	00870				:	
					54	4E	43	2F	21	29	58	28	57	58	0087F				:	
58	28	57	58	21	20	3A	32	20	47	45	52	20	4C	52	00889		.ASCII	\RL REG 2: !XW(X)\	:	
														29	00898				:	
50	4D	55	44	20	52	53	43	20	49	42	55	2F	21	4B	00899	P.ABW:	.ASCII	\K!/UBI CSR DUMP:!/CSR 0: !XL(X)!/CSR 1: \	:	
58	28	4C	58	21	20	3A	30	20	52	53	43	2F	21	3A	008A8				:	
					20	3A	31	20	52	53	43	2F	21	29	008B7				:	
20	3A	32	20	52	53	43	2F	21	29	58	28	4C	58	21	008C1		.ASCII	\!XL(X)!/CSR 2: !XL(X)!/CSR 3: !XL(X)\	:	
20	3A	33	20	52	53	43	2F	21	29	58	28	4C	58	21	008D0				:	
										29	58	28	4C	58	21	008DF				:
4C	43	41	20	31	52	53	43	20	54	45	55	2F	21	4A	008E5	P.ABX:	.ASCII	\J!/UET CSR1 ACLOW1 NOT IN CORRECT STATE!\	:	
52	52	4F	43	20	4E	49	20	54	4F	4E	20	31	57	4F	008F4				:	
					21	45	54	41	54	53	20	54	43	45	00903				:	
28	57	58	21	20	3A	44	45	54	43	45	50	58	45	2F	0090D		.ASCII	\!/EXPECTED: !XW(X)!/RECEIVED: !XW(X)\	:	
21	20	3A	44	45	56	49	45	43	45	52	2F	21	29	58	0091C				:	
										29	58	28	57	58	0092B				:	
54	4E	49	20	31	52	53	43	20	54	45	55	2F	21	4B	00930	P.ABY:	.ASCII	\K!/UET CSR1 INTERRUPT DONE BIT INCORRECT\	:	
54	49	42	20	45	4E	4F	44	20	54	50	55	52	52	45	0093F				:	
					54	43	45	52	52	4F	43	4E	49	20	0094E				:	
57	58	21	20	3A	44	45	54	43	45	50	58	45	2F	21	00958		.ASCII	\!/EXPECTED: !XW(X)!/RECEIVED: !XW(X)\	:	
20	3A	44	45	56	49	45	43	45	52	2F	21	29	58	28	00967				:	
										29	58	28	57	58	21	00976				:
54	4E	49	20	32	52	53	43	20	54	45	55	2F	21	4B	0097C	P.ABZ:	.ASCII	\K!/UET CSR2 INTERRUPT DONE BIT INCORRECT\	:	
54	49	42	20	45	4E	4F	44	20	54	50	55	52	52	45	0098B				:	
					54	43	45	52	52	4F	43	4E	49	20	0099A				:	
57	58	21	20	3A	44	45	54	43	45	50	58	45	2F	21	009A4		.ASCII	\!/EXPECTED: !XW(X)!/RECEIVED: !XW(X)\	:	
20	3A	44	45	56	49	45	43	45	52	2F	21	29	58	28	009B3				:	
										29	58	28	57	58	21	009C2				:
45	55	20	4E	49	20	54	49	42	20	42	50	2F	21	2C	009C8	P.ACA:	.ASCII	\,!/PB BIT IN UET DID NOT CLEAR WITH UET \	:	
52	41	45	4C	43	20	54	4F	4E	20	44	49	44	20	54	009D7				:	
					20	54	45	55	20	48	54	49	57	20	009E6				:	
										2E	54	49	4E	49	009F0		.ASCII	\INIT.\	:	
45	20	53	55	54	41	54	53	20	54	45	55	2F	21	36	009F5	P.ACB:	.ASCII	\6!/UET STATUS ERROR!/EXPECTED: !XW(X)!/R\	:	
3A	44	45	54	43	45	50	58	45	2F	21	52	4F	52	52	00A04				:	
					52	2F	21	29	58	28	57	58	21	20	00A13				:	
29	58	28	57	58	21	20	3A	44	45	56	49	45	43	45	00A1D		.ASCII	\ECEIVED: !XW(X)\	:	
55	42	49	4E	55	20	4F	54	20	55	50	43	2F	21	2A	00A2C	P.ACC:	.ASCII	\!/CPU TO UNIBUS TIMEOUT AT ADDRESS: !XL\	:	
44	41	20	54	41	20	54	55	4F	45	4D	49	54	20	53	00A3B				:	

					4C	58	21	20	3A	53	53	45	52	44	00A4A			
55	20	44	45	54	43	45	50	58	45	4E	55	29	58	28	00A54	P.ACD:	.ASCII	\(X)\
43	4F	20	54	50	55	52	52	45	54	4E	55	2F	21	2D	00A57		.ASCII	\-!/UNEXPECTED UET INTERRUPT OCCURRED AT \
					20	54	41	20	44	45	52	52	55	43	00A66			
									4C	58	21	3A	43	50	00A75			
49	20	44	45	54	43	45	50	58	45	4E	55	2F	21	2F	00A7F	P.ACE:	.ASCII	\PC:!XL\
21	52	42	20	54	41	20	54	50	55	52	52	45	54	4E	00A85		.ASCII	\!/UNEXPECTED INTERRUPT AT BR!UB WITH IP\
					50	49	20	48	54	49	57	20	42	55	00A94			
							42	55	21	20	54	41	20	4C	00AA3			
41	20	44	45	54	43	45	50	58	45	4E	55	2F	21	35	00AAD	P.ACF:	.ASCII	\L AT !UB\
20	54	50	55	52	52	45	54	4E	49	20	57	4F	4C	43	00AB5		.ASCII	\5!/UNEXPECTED ACLOW INTERRUPT AT BR!UB W\
					57	20	42	55	21	52	42	20	54	41	00AC4			
	42	55	21	20	54	41	20	4C	50	49	20	48	54	49	00AD3			
41	20	44	45	54	43	45	50	58	45	4E	55	2F	21	38	00ADD	P.ACG:	.ASCII	\ITH IPL AT !UB\
20	54	50	55	52	52	45	54	4E	49	20	57	4F	4C	43	00AEB		.ASCII	\8!/UNEXPECTED ACLOW INTERRUPT WITH INTER\
					52	45	54	4E	49	20	48	54	49	57	00AFA			
45	4C	43	20	45	4C	42	41	4E	45	20	54	50	55	52	00B09		.ASCII	\RUPT ENABLE CLEAR\
													52	41	00B13			
44	49	44	20	54	45	53	45	52	20	4F	49	2F	21	27	00B22	P.ACH:	.ASCII	\'!/IO RESET DID NOT CLEAR ACLOW1 IN CSR1\
4F	4C	43	41	20	52	41	45	4C	43	20	54	4F	4E	20	00B24			
					31	52	53	43	20	4E	49	20	31	57	00B33			
4E	49	20	54	43	45	52	52	4F	43	4E	49	2F	21	40	00B42	P.ACI:	.ASCII	\a!/INCORRECT INTERRUPT VECTOR!/EXPECTED:\
21	52	4F	54	43	45	56	20	54	50	55	52	52	45	54	00B4C			
					3A	44	45	54	43	45	50	58	45	2F	00B5B			
56	49	45	43	45	52	2F	21	29	58	28	57	58	21	20	00B6A		.ASCII	\ !XW(X)!/RECEIVED: !XW(X)\
					29	58	28	57	58	21	20	3A	44	45	00B74			
		20	3A	44	45	56	49	45	43	45	52	20	0B	00B83				
							20	3A	52	4F	58	20	06	00B8D	P.ACJ:	.ASCII	<11>\ RECEIVED: \	
4C	58	21	20	3A	44	45	54	43	45	50	58	45	20	39	00B99	P.ACK:	.ASCII	<6>\ XOR: \
3A	44	45	56	49	45	43	45	52	20	2F	21	29	58	28	00BA0	P.ACL:	.ASCII	\9 EXPECTED: !XL(X)!/ RECEIVED: !XL(X)!/ \
					20	2F	21	29	58	28	4C	58	21	20	00BAF			
58	28	4C	58	21	20	20												

8
3)

ZZ-ECCBA-1.4
UBI HEADER
01-04

INITIALIZE
INITIALIZE

F 6

6-Sep-1985

Fiche 1

Frame F6

Sequence 70

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 21
(4)

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21

```
55 20 3A 73 43 49 20 47 4E 49 4C 49 41 46 13 00CB6 P.ACX: .ASCII <19>\FAILING ICs: UDP0-3\
      20 3A 73 43 49 20 47 4E 49 4C 49 41 46 0D 00CC5
      20 3A 73 43 49 20 47 4E 49 4C 49 41 46 0D 00CCA P.ACY: .ASCII <13>\FAILING ICs: \
45 4C 55 44 4F 4D 20 47 4E 49 4C 49 41 46 13 00CD8 P.ACZ: .ASCII <13>\FAILING ICs: \
      54 45 55 20 3A 00CE6 P.ADA: .ASCII <19>\FAILING MODULE: UET\
      54 45 55 20 3A 00CF5
4E 55 2C 45 43 55 2C 4D 58 4E 2C 52 52 45 20 00CFA P.ADB: .ASCII \ ERR,NXM,UCE,UNDEFINED=28^X,PURGE\
55 50 2C 58 5E 38 32 3D 44 45 4E 49 46 45 44 00D09
      45 47 52 00D18
44 5F 4E 4F 5F 42 4F 54 41 44 2C 52 52 45 5C 00D1B P.ADC: .ASCII <92>\ERR,DATOB_ON_DATIP,INH_DATIP_ROL,C\
5F 50 49 54 41 44 5F 48 4E 49 2C 50 49 54 41 00D2A
      43 2C 4C 4F 52 00D39
4E 55 46 2C 42 4E 55 46 2C 41 54 41 44 2B 43 00D3E .ASCII \C+DATA,FUNB,FUNA,C1,C0,RDY,INT_ODNE,NPR,\
54 4E 49 2C 59 44 52 2C 30 43 2C 31 43 2C 41 00D4D
      2C 52 50 4E 2C 45 4E 44 4F 5F 00D5C
34 52 42 2C 35 52 42 2C 36 52 42 2C 37 52 42 00D66 .ASCII \BR7,BR6,BR5,BR4,G0\
      4F 47 2C 00D75
43 2C 59 4C 44 5F 4D 54 2C 35 31 54 49 42 A5 00D78 P.ADD: .ASCII <165>\BIT15,TM_DLY,CCOVF,ENB_PB,INTR_SSY\
54 4E 49 2C 42 50 5F 42 4E 45 2C 46 56 4F 43 00D87
      59 53 53 5F 52 00D96
54 4F 4E 2B 54 4E 47 4F 4E 2C 52 52 45 5F 4E 00D9B .ASCII \N_ERR,NOGNT+NOTONEGNT,WRG_A_LINES,NO_SSY\
49 4C 5F 41 5F 47 52 57 2C 54 4E 47 45 4E 4F 00DAA
      59 53 53 5F 4F 4E 2C 53 45 4E 00DB9
43 41 53 5F 4F 4E 5F 4F 4E 2C 52 52 45 5F 4E 00DC3 .ASCII \N_ERR,NO_NO_SACK_ERR,MAX_LATE_ERR,WNG_GN\
5F 45 54 41 4C 5F 58 41 4D 2C 52 52 45 5F 4B 00DD2
      4E 47 5F 47 4E 57 2C 52 52 45 00DE1
5F 4E 57 44 5F 52 57 50 2C 4B 43 41 42 5F 54 00DEB .ASCII \T_BACK,PWR_DWN_SEQ,INH_SACK,INH_INC_ADDR\
4E 49 2C 4B 43 41 53 5F 48 4E 49 2C 51 45 53 00DFA
      52 44 44 41 5F 43 4E 49 5F 48 00E09
      36 31 41 2C 37 31 41 2C 43 43 5F 00E13 .ASCII \_CC,A17,A16\
```

.PSECT \$OWN\$,NOEXE,2

00000 DEV_REG:.BLKB 0

```
SECTION= P.AAA-4
AL_DEVTYPE= P.AAE-4
DSA$AL_APTMAIL= 65024
DSA$GL_FLAGS= 65024
DSA$GL_APTCOM= 65028
DSA$GL_PASSES= 65032
DSA$GL_UNITS= 65036
DSA$GL_SECTNO= 65040
DSA$GL_SID= 65044
DSA$GL_MSGTYP= 65088
DSA$GL_ERRNO= 65092
DSA$GL_EVENT= 65096
DSA$GL_SUBTNO= 65100
DSA$GL_TESTNO= 65104
DSA$GL_PASSNO= 65108
DSA$GL_DEVLEN= 65112
DSA$GL_DEVNAM= 65116
DSA$GL_MSGPTR= 65128
EXP_MSG== P.AAI
FMT_BODY== P.AAJ
```

9
3)
ZZ-ECCBA-1.4
UBI_HEADER
01-04

INITIALIZE
INITIALIZE

G 6

6-Sep-1985

Fiche 1

Frame G6

Sequence 71

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 22

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21

(4)

FMT_BDP_DATI== P.AAK
FMT_BDP_DATO== P.AAL
FMT_BDP_DATO== P.AAM
FMT_BYTE_OFF== P.AAN
FMT_CMI_UB== P.AAO
FMT_CPU_RW== P.AAP
FMT_CSR_ERR1== P.AAQ
FMT_CSR_ERR2== P.AAR
FMT_DDP_DATI== P.AAS
FMT_DDP_DATO== P.AAT
FMT_DDP_DATO== P.AAU
FMT_DP_SEL== P.AAV
FMT_DCMP_ERR1== P.AAW
FMT_DCMP_ERR1A== P.AAX
FMT_DCMP_ERR2== P.AAY
FMT_EXT_PROC== P.AAZ
FMT_INTERLOCK== P.ABA
FMT_INIT_ACLO== P.ABB
FMT_MAP_ADDR== P.ABC
FMT_MAP_BIT_SA0== P.ABD
FMT_MAP_BIT_SA1== P.ABE
FMT_MAP_GEN== P.ABF
FMT_MAP_CS== P.ABG
FMT_MAP_DB_SA0== P.ABH
FMT_MAP_DB_SA1== P.ABI
FMT_MAP_FAIL== P.ABJ
FMT_MAP_INV== P.ABK
FMT_MCHK== P.ABL
FMT_NO_INT== P.ABM
FMT_NO_ACINT== P.ABN
FMT_NO_ACINT1== P.ABO
FMT_NO_MCHK== P.ABP
FMT_NO_UB== P.ABQ
FMT_NO_UBI== P.ABR
FMT_NOTHING== P.ABS
FMT_UA1== P.ABT
FMT_UB_CMI== P.ABU
FMT_UB_DUMP== P.ABV
FMT_UBI_DUMP== P.ABW
FMT_UET_ACLO1== P.ABX
FMT_UET_INTD== P.ABY
FMT_UET_INTD2== P.ABZ
FMT_UET_PB== P.ACA
FMT_UET_STAT== P.ACB
FMT_UB_TO== P.ACC
FMT_UNX_INT== P.ACD
FMT_UNX_INT1== P.ACE
FMT_UNX_ACINT== P.ACF
FMT_UNX_ACINT1== P.ACG
FMT_UNX_ACINT2== P.ACH
FMT_VECTOR== P.ACI
REC_MSG== P.ACJ
XOR_MSG== P.ACK
EXP_REC_XOR== P.ACL
UBIMOD== P.ACM

10
3)
ZZ-ECCBA-1.4
UBI HEADER
01-04
INITIALIZE
INITIALIZE

H 6
6-Sep-1985 6-Sep-1985 6-Sep-1985
Fiche 1 17:17:45 17:14:42
Frame H6
Sequence 72
VAX-11 Bliss-32 V4.1-746
[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21
Page 23
(4)

UBIMOD_BDP== P.ACN
UBIMOD_BYTE_OFF== P.ACO
UBIMOD_CMI== P.ACP
UBIMOD_CPU_RW== P.ACQ
UBIMOD_CSR== P.ACR
UBIMOD_DDP== P.ACS
UBIMOD_DP_SEL== P.ACT
UBIMOD_MAP== P.ACU
UBIMOD_MAP_ADDR== P.ACV
UBIMOD_MAP_CS== P.ACW
UBIMOD_MAP_CTRL== P.ACX
UBIMOD_UA1== P.ACY
UBIMOD_UB_CMI== P.ACZ
UETMOD== P.ADA
CSR_MNEM== P.ADB
UBECR1_MNEM== P.ADC
UBECR2_MNEM== P.ADD
.EXTRN UBE0_ISR, UBE1_ISR
.EXTRN UBE2_ISR, UBE3_ISR
.EXTRN DS\$GPHARD, DS\$PRINTX
.EXTRN DS\$ABORT, DS\$ENDPASS
.EXTRN DS\$SETVEC, DS\$BREAK

.PSECT INITIALIZE,2

56 00000000G 007C 00000
55 0000' 9F 9E 00002
5E 10 C2 0000E
50 01 CE 00011
37 50 DA 00014

.ENTRY INITIALIZE, Save R2,R3,R4,R5,R6 ; 0455
MOVAB a#DS\$GPHARD, R6 ;
MOVAB PTBASE, R5 ;
SUBL2 #16, SP ;
MNEGL #1, TEMP ; 0493
MTPR TEMP, #55 ; 0494

; One p-table is associated with each logical number (LN). The
; user
; attach and select UBIs and UBEs in any order. Therefore, p-t
; ables for UBIs
; and UBEs may be in any order. Because of this, it is necessa
; ry to
; determine the device type for each p-table to ascertain what
; action is to
; be taken with a particular p-table.
; All existing p-tables are scanned, looking for UBIs (DW750s).
; The first
; UBI found is assigned as UBI logical number 0. After all UBI
; s are found
; and assigned logical numbers, the p-tables are again searched
; for UBEs.
; Each UBE which is associated with the current UBI under test
; is assigned
; with a logical number. The UBE address is stored in the vect
; or
; UBE_BASE_ADDR and its vector is captured.

03 0000FE03 9F 05 E0 00017 BBS #5, a#^X0000FE03, 1\$; 0517

			0084	31	0001F	BRW	7\$		
D2	A5		01	8E	00022	MNEGB	#1, LUN	:	0520
	54	45425520	8F	D0	00026	MOVL	#1161975072, UBE	:	0522
	54		03	90	0002D	MOVB	#3, UBE	:	0523
		C8	A5	D4	00030	CLRL	NUM_UBI	:	0525
			55	DD	00033	PUSHL	R5	:	0527
	7E		A5	9A	00035	MOVZBL	LN, -(SP)	:	
	66		02	FB	00039	CALLS	#2, DS\$GPHARD	:	
	21		50	E9	0003C	BLBC	R0, 5\$	:	
	50		65	D0	0003F	MOVL	PTBASE, R0	:	0529
	54	26	A0	D1	00042	CMPL	38(R0), UBE	:	
			06	12	00046	BNEQ	3\$	:	
C9	A5		01	90	00048	MOVB	#1, UBE_SEL	:	0531
			0D	11	0004C	BRB	4\$	:	
	50	C8	A5	9A	0004E	MOVZBL	NUM_UBI, R0	:	0534
8C	A540	CB	A5	9A	00052	MOVZBL	LN, UBI_LN[R0]	:	
		C8	A5	96	00058	INCB	NUM_UBI	:	0535
		CB	A5	96	0005B	INCB	LN	:	0537
			D3	11	0005E	BRB	2\$	:	0527
		C8	A5	95	00060	TSTB	NUM_UBI	:	0540
			12	12	00063	BNEQ	6\$	:	
		0000'	CF	9F	00065	PUSHAB	FMT_NO_UBI	:	0543
00000000G	9F		01	FB	00069	CALLS	#1, @DS\$PRINTX	:	
00000000G	9F		00	FB	00070	CALLS	#0, @DS\$ABORT	:	0544
CA	A5	CB	A5	9B	00077	MOVZBW	LN, DEVICES	:	0547
		FC	A5	D4	0007C	CLRL	NUM_UBE	:	0549
0454	C5	00000000G	EF	9E	0007F	MOVAB	UBE0_ISR, UBE_SERVICE	:	0550
0458	C5	00000000G	EF	9E	00088	MOVAB	UBE1_ISR, UBE_SERVICE+4	:	0551
045C	C5	00000000G	EF	9E	00091	MOVAB	UBE2_ISR, UBE_SERVICE+8	:	0552
0460	C5	00000000G	EF	9E	0009A	MOVAB	UBE3_ISR, UBE_SERVICE+12	:	0553
		BD	A5	94	000A3	CLRB	EXTERNAL_FLAG	:	0554
		D2	A5	96	000A6	INCB	LUN	:	0559
C8	A5	D2	A5	91	000A9	CMPB	LUN, NUM_UBI	:	0560
			0A	1F	000AE	BLSSU	8\$	:	
00000000G	9F		00	FB	000B0	CALLS	#0, @DS\$ENDPASS	:	0562
		D2	A5	94	000B7	CLRB	LUN	:	0564
			55	DD	000BA	PUSHL	R5	:	0568
	50	D2	A5	9A	000BC	MOVZBL	LUN, R0	:	
		8C	A540	DD	000C0	PUSHL	UBI_LN[R0]	:	
	66		02	FB	000C4	CALLS	#2, DS\$GPHARD	:	
	50		65	D0	000C7	MOVL	PTBASE, R0	:	0569
84	A5	18	A0	D0	000CA	MOVL	24(R0), UBI_UNDER_TEST	:	
94	A5	1C	A0	D0	000CF	MOVL	28(R0), UNIBUS_SPACE	:	0570
0488	C5		50	D0	000D4	MOVL	R0, UBI_LINK	:	0571
88	A5	24	A0	B0	000D9	MOVW	36(R0), UBI_VECTOR	:	0572
	54	C9	A5	E9	000DE	BLBC	UBE_SEL, 11\$	:	0575
	53	CA	A5	9A	000E2	MOVZBL	DEVICES, R3	:	0578
	52		01	CE	000E6	MNEGL	#1, LN	:	0581
			47	11	000E9	BRB	10\$	:	
			24	BB	000EB	PUSHR	#^M<R2,R5>	:	0580
	66		02	FB	000ED	CALLS	#2, DS\$GPHARD	:	
	51		65	D0	000F0	MOVL	PTBASE, R1	:	0581
	54	26	A1	D1	000F3	CMPL	38(R1), UBE	:	
			39	12	000F7	BNEQ	10\$	:	
0488	C5	20	A1	D1	000F9	CMPL	32(R1), UBI_LINK	:	0584

12
4)

ZZ-ECCBA-1.4
UBI_HEADER
01-04

INITIALIZE
INITIALIZE

J 6

6-Sep-1985

Fiche 1 Frame J6

Sequence 74

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 25

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21

(4)

047C C540 08 A540 24 A1 0484 C540
00000000G 7E 9F
B5 00000000G 52 9F
FC 31 12 000FF
A5 D0 00101
A1 D0 00105
A5 A9 0010B
A1 90 00114
7E D4 0011B
0454 C540 DD 0011D
047C C540 3C 00122
03 FB 00128
FC A5 D6 0012F
53 F2 00132 10\$:
00 FB 00136 11\$:
04 0013D

BNEQ 10\$;
MOVL NUM_UBE, R0 ; 0587
MOVL 24(R1), UBE_BASE_ADDR[R0] ;
BISW3 UBI_VECTOR, 36(R1), UBE_VECTOR[R0] ; 0588
MOVB 11(R1), UBE_DRIVE[R0] ; 0589
CLRL -(SP) ; 0591
PUSHL UBE_SERVICE[R0] ;
MOVZWL UBE_VECTOR[R0], -(SP) ;
CALLS #3, a#DS\$SETVEC ;
INCL NUM_UBE ; 0592
AOBLSS R3, LN, 9\$; 0578
CALLS #0, a#DS\$BREAK ; 0597
RET ; 0599

; Routine Size: 318 bytes, Routine Base: INITIALIZE + 0000

3)
ZZ-ECCBA-1.4
UBI_HEADER
01-04

CLEAN UP

CLEAN UP

K 6

6-Sep-1985

Fiche 1

Frame K6

Sequence 75

6-Sep-1985

17:17:45

VAX-11 Bliss-32 V4.1-746

Page 26

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21

(5)

```
: 0600 2 $DS_BGNCLEAN ;
: 0601 2 !+
: 0602 2 ! FUNCTIONAL DESCRIPTION:
: 0603 2 !
: 0604 2 !     Not clear what this routine should do at this time.
: 0605 2 !
: 0606 2 ! FORMAL PARAMETERS: NONE
: 0607 2 !
: 0608 2 ! IMPLICIT INPUTS: NONE
: 0609 2 !
: 0610 2 ! IMPLICIT OUTPUTS: NONE
: 0611 2 !
: 0612 2 ! COMPLETION CODES: NONE
: 0613 2 !
: 0614 2 ! SIDE EFFECTS:
: 0615 2 !
: 0616 2 !     All selected UBIs are left in a clean state.
: 0617 2 ! --
: 0618 2
: 0619 2 LOCAL
: 0620 2     TEMP,
: 0621 2     MEM_CSR: REF VECTOR[3];
: 0622 2
: 0623 2 LITERAL
: 0624 2     IO_RESET=  &X'37',
: 0625 2     CHC$_DSINT = 5;
: 0626 2
: 0627 2 BUILTIN
: 0628 2     MTPR;
: 0629 2
: 0630 2     TEMP = -1;
: 0631 2     MTPR(TEMP,IO_RESET);
: 0632 2
: 0633 2     $DS_CHANNEL(UNIT=.LUN,FUNC=CHC$_DSINT);
: 0634 2     $DS_INITSCB;
: 0635 2
: 0636 2 !+
: 0637 2 ! Cleanup the memory controller
: 0638 2 !-
: 0639 2     MEM_CSR = MEMORY_CONT;
: 0640 2     MEM_CSR[0] = CLEAR_MEM_ERR;
: 0641 2     MEM_CSR[1] = 0;
: 0642 2
: 0643 1 $DS_ENDCLEAN ;
```

.EXTRN DS\$CHANNEL, DS\$INITSCB

.PSECT CLEANUP,2

.ENTRY CLEAN_UP, Save nothing

MNEGL #1, TEMP

MTPR TEMP, #55

CLRL @#^X00000000

50
37

00000000

```
0000 00000
01 CE 00002
50 DA 00005
9F D4 00008
```

```
: 0599
: 0630
: 0631
: 0633
```


4
1)
ZZ-ECCBA-1.4
UBI_HEADER
01-04
CLEAN UP
CLEAN UP

L 6
6-Sep-1985
6-Sep-1985 17:17:45
6-Sep-1985 17:14:42
Fiche 1 Frame L6
VAX-11 Bliss-32 V4.1-746
[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21
Sequence 76
Page 27
(5)

	7E		7E	7C	0000E	CLRQ	-(SP)	:
	7E		05	7D	00010	MOVQ	#5, -(SP)	:
	7E	0000'	CF	9A	00013	MOVZBL	LUN, -(SP)	:
00000000G	9F		05	FB	00018	CALLS	#5, a#DS\$CHANNEL	:
00000000G	9F		00	FB	0001F	CALLS	#0, a#DS\$INITSCB	:
	50	00F20000	8F	D0	00026	MOVL	#15859712, MEM_CSR	: 0639
	60	E0000000	8F	D0	0002D	MOVL	#-536870912, (MEM_CSR)	: 0640
		04	A0	D4	00034	CLRL	4(MEM_CSR)	: 0641
				04	00037	RET		: 0643

; Routine Size: 56 bytes, Routine Base: CLEANUP + 0000

5)

SUMMARY

SUMMARY

```

: 0644 2 $SDS_BGN SUMMARY ;
: 0645 2 !++
: 0646 2 ! FUNCTIONAL DESCRIPTION:
: 0647 2 !
: 0648 2 ! FORMAL PARAMETERS: NONE
: 0649 2 !
: 0650 2 ! IMPLICIT INPUTS: NONE
: 0651 2 !
: 0652 2 ! IMPLICIT OUTPUTS: NONE
: 0653 2 !
: 0654 2 ! COMPLETION CODES: NONE
: 0655 2 !
: 0656 2 ! SIDE EFFECTS:
: 0657 2 !
: 0658 2 !--
: 0659 1 $SDS_END SUMMARY ;

```

M 6

6-Sep-1985

Fiche 1 Frame M6

Sequence 77

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 28

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32:21

(6)

.PSECT SUMMARY,2

000000000G 9F

```

0000 00000
00 FB 00002
    04 00009

```

```
.ENTRY SUMMARY, Save nothing
CALLS #0, a#DS$BREAK
RET
```

: 0643
: 0644
: 0659

; Routine Size: 10 bytes, Routine Base: SUMMARY + 0000

6
)
ZZ-ECCBA-1.4
UBI HEADER
01-04

SUMMARY

SUMMARY

; 0660 1 \$DS_ENDMOD ;
; 0661 1 END
; 0662 0 ELUDOM

!End of module

N 6

6-Sep-1985

Fiche 1

Frame N6

Sequence 78

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 29

6-Sep-1985 17:14:42

[LEMIEUX.ECCBA.RELEASE]ECCBA0.B32;21

(7)

.PSECT \$TSTCNT,NOWRT,NOEXE, OVR,2

00000 L_TSTCNT:

.BLKB 4

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	0	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
DISPATCH	0	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
DISPATCH_X	24	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	3614	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
LAST	0	NOVEC, WRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$HEADER	88	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(9)
GLOBALS	1386	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
SCRATCHPAD	512	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(9)
INITIALIZE	318	NOVEC, WRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
CLEANUP	56	NOVEC, WRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
SUMMARY	10	NOVEC, WRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$TSTCNT	4	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, OVR, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
SYS\$COMMON:[SYSLIB]DIAG.L32;265	784	49	6	85	00:00.1
SYS\$COMMON:[SYSLIB]STARLET.L32;3	10093	0	0	598	00:01.4

COMMAND QUALIFIERS

; BLISS/LIST ECCBA0.B32

; Size: 384 code + 5628 data bytes
; Run Time: 00:13.3
; Elapsed Time: 00:19.2

ZZ-ECCBA-1.4 SUMMARY
UBI HEADER
01-04 SUMMARY

; Lines/CPU Min: 2995
; Lexemes/CPU-Min: 26692
; Memory Used: 217 pages
; Compilation Complete

B 7

6-Sep-1985

Fiche 1 Frame B7

Sequence 79

6-Sep-1985 17:17:45

VAX-11 Bliss-32 V4.1-746

Page 30

```

: 0001 0  MODULE UBI_SUBROUTINES (      ! UBI Diagnostic Program subroutines
: 0002 0      IDENT = '01-04'
: 0003 0      ) =
: 0004 1  BEGIN
: 0005 1
: 0006 1  !
: 0007 1  ! COPYRIGHT (C) 1980,1981
: 0008 1  ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
: 0009 1  !
: 0010 1  ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
: 0011 1  ! COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
: 0012 1  ! ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
: 0013 1  ! MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
: 0014 1  ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
: 0015 1  ! TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
: 0016 1  ! REMAIN IN DEC.
: 0017 1  !
: 0018 1  ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
: 0019 1  ! AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
: 0020 1  ! CORPORATION.
: 0021 1  !
: 0022 1  ! DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
: 0023 1  ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
: 0024 1  !
: 0025 1
: 0026 1  !**
: 0027 1  ! FACILITY:      VAX-11 Diagnostic Library
: 0028 1
: 0029 1  ! ABSTRACT:      This module contains the subroutines used by the
: 0030 1  !                  COMET Unibus Interface Diagnostic Program.
: 0031 1
: 0032 1  ! ENVIRONMENT:    VAX-11 Diagnostic Supervisor
: 0033 1
: 0034 1  ! AUTHOR:          Rand Gray, CREATION DATE: 28-Nov-78
: 0035 1
: 0036 1  ! MODIFIED BY:     Don Monroe
: 0037 1
: 0038 1  !           0-7      May 1979
: 0039 1  !                  Changed the format of the UBI MAP registers.
: 0040 1
: 0041 1  !           0-8      July 1979
: 0042 1  !                  Modified to support the real UET module.
: 0043 1
: 0044 1  !                  Fixed a BUG in XFER_SETUP routine. Loop that setup the maps
: 0045 1  !                  would not work if MAP_NUM was not zero.
: 0046 1
: 0047 1  !           0-9      July 1979
: 0048 1  !                  Changed references to 'Byte Offset' to 'Byte Number'
: 0049 1
: 0050 1
: 0051 1  !           0-10     October 1979
: 0052 1  !                  Added a return value to the MAP_BUFFERS routine to support
: 0053 1  !                  the changes to Module 7 Revision 10.
: 0054 1  !                  Added byte-offset parameter to MAP_BUFFERS routine so routine
: 0055 1  !                  returns with one additional page per buffer.

```

```

: 0056 1 !
: 0057 1 !
: 0058 1 ! 0-11 October 22,1979
: 0059 1 ! Added a routine to print MAP errors. Called via PRINT LINK
: 0060 1 ! in error hard calls.
: 0061 1 !
: 0062 1 ! 0-12 December 7,1979
: 0063 1 ! Fixed bug in PROBE_ADDRESS routine in the ADAWI builtin.
: 0064 1 !
: 0065 1 ! 0-13 April 7,1980
: 0066 1 ! Changed address of unibus space so second UBI will work.
: 0067 1 !
: 0068 1 ! 0-14 April 8,1980
: 0069 1 ! Added a print general routine and converted PRINTX after
: 0070 1 ! error calls to print links.
: 0071 1 ! 1-00 June 16,1980
: 0072 1 ! Initial Release
: 0073 1 ! 1-01 July 31,1980
: 0074 1 ! Fixed bug in MAP_BUFFERS routine that would not detect
: 0075 1 ! insufficient memory for the requested number of buffers.
: 0076 1 ! 1-02 December 29,1980
: 0077 1 ! Fixed bug in MAP_BUFFERS routine. Supervisor allocates memory
: 0078 1 ! even if not enough for requested buffer size.
: 0079 1 ! 1-03 May 6, 1981
: 0080 1 ! Fixed bug in map_buffers routine.
: 0081 1 !
: 0082 1 ! Kevin Lemieux
: 0083 1 !
: 0084 1 ! 1-04 February,1985
: 0085 1 ! Fixed bug in INIT code. If two DW's were specified; after the
: 0086 1 ! first pass one of them one of them was no longer tested.
: 0087 1 ! Fixed minor bugs in TEST 29.
: 0087 1 ! --

```



```

: 0088 1 !
: 0089 1 ! TABLE OF CONTENTS:
: 0090 1 !
: 0091 1 !
: 0092 1 FORWARD ROUTINE
: 0093 1 PRINT_CSR_ERR:NOVALUE, ! Prints a configuration/status error
: 0094 1 PRINT_DCMF_ERR:NOVALUE, ! Prints a data compare error
: 0095 1 PRINT_MAP_ERR:NOVALUE, ! Prints a MAP register error A11
: 0096 1 PRINT_GENERAL:NOVALUE, ! Prints a general error report A14
: 0097 1 HANDLER, ! Condition handler routine
: 0098 1 UBE_INTERRUPT:NOVALUE, ! Interrupt service routine for UBEs
: 0099 1 XFER_ANALYSIS:NOVALUE, ! Transfer analysis routine
: 0100 1 PROBE_ADDRESS, ! Routine to see if address exists
: 0101 1 XFER_SETUP:NOVALUE, ! Transfer setup routine
: 0102 1 MAP_SETUP:NOVALUE, ! UBI map setup routine
: 0103 1 BUFFER_SETUP:NOVALUE, ! Sets up I/O buffer for UBE transfer
: 0104 1 MAP_BUFFERS, ! Allocates free memory for buffer use M10
: 0105 1 UNIBUS_SIZER:NOVALUE, ! Sizer for Unibus memory
: 0106 1 ENCODER:NOVALUE, ! Creates Reed-Muller (16,5) codewords
: 0107 1 DECODER:NOVALUE, ! Decodes possibly corrupt codewords
: 0108 1
: 0109 1 !
: 0110 1 ! INCLUDE FILES:
: 0111 1 !
: 0112 1
: 0113 1 LIBRARY 'SYS$LIBRARY:STARLET';
: 0114 1 LIBRARY 'SYS$LIBRARY:DIAG' ;
: 0115 1
: 0116 1 !
: 0117 1 ! DIAGNOSTIC SUPERVISOR MACROS:
: 0118 1 !
: 0119 1
: L 0120 1 $DS_BGNMOD(ENV=CEP_REPAIR);
: xPRINT: 1 Bliss-32 Diagnostic Macro Library version 7.0 "DIAG.L32 (57)"
: 0121 1 $DS_DSADEF;
: 0122 1
: 0123 1 !
: 0124 1 ! EQUATED SYMBOLS:
: 0125 1 !
: 0126 1
: 0127 1 LITERAL
: 0128 1 CSR_MASK= xX'E0000001', ! Mask for UBI CSR's
: 0129 1 MAP_MASK= xX'82607FFF', ! Mask for map registers A11
: 0130 1 ERROR= BIT15, ! UBE CR1 bit 15 (composite error)
: 0131 1 UBI_MAP_OFFSET= xX'800', ! Offset to UBI map entries
: 0132 1 ! space
: 0133 1 !
: 0134 1 ! OWN STORAGE:
: 0135 1 !
: 0136 1
: 0137 1 !
: 0138 1 ! EXTERNAL REFERENCES:
: 0139 1 !
: 0140 1
: 0141 1 STRUCTURE

```

```

: 0142 1      ERROR_BUFFER[I, 0; N, BS, UNIT=4] =
: 0143 1      [N*BS*UNIT]
: 0144 1      (ERROR_BUFFER+(0+I*BS)*UNIT);
: 0145 1
: 0146 1      EXTERNAL
: 0147 1      CODEWORDS: VECTOR[9,WORD],
: 0148 1      CODEVECTORS: VECTOR[4,WORD],
: 0149 1      CSR_MNEM,
: 0150 1      DECODED_WORDS: VECTOR[9],
: 0151 1      EVENT_FLAG: BITVECTOR[4],
: 0152 1      EXP_MSG,
: 0153 1      EXP_REC_XOR,
: 0154 1      FMT_BODY,
: 0155 1      FMT_CSR_ERR1,
: 0156 1      FMT_CSR_ERR2,
: 0157 1      FMT_DCMP_ERR1,
: 0158 1      FMT_DCMP_ERR1A,
: 0159 1      FMT_DCMP_ERR2,
: 0160 1      FMT_MAP_GEN,
: 0161 1      FMT_NOTHING,
: 0162 1      FREE_MAP: VECTOR[4,WORD],
: 0163 1      LUN: BYTE,
: 0164 1      NOISY_WORDS: VECTOR[9,WORD],
: 0165 1      NUM_MAPS: WORD,
: 0166 1      NUM_DATA_ERR: VECTOR[4],
: 0167 1      REC_MSG,
: 0168 1      UBE_BASE_ADDR: VECTOR[4],
: 0169 1      UBE_BUF_ADDR: VECTOR[4],
: 0170 1      UBE_MODE: VECTOR[4,BYTE],
: 0171 1      UBE_MAP_NO: VECTOR[4,WORD],
: 0172 1      UBE_DP_NO: VECTOR[4,BYTE],
: 0173 1      UBE_BUF_SIZE,
: 0174 1      UBE_PAGE_CNT,
: 0175 1      UBE_ERR_BUF: ERROR_BUFFER[4,64],
: 0176 1      UBE_EXP_DATA: VECTOR[4,WORD],
: 0177 1      UBE_INIT_DATA: VECTOR[4,WORD],
: 0178 1      UBE_INT_OCC: VECTOR[4,BYTE],
: 0179 1      UBE_XFER_ERR: BITVECTOR[4],
: 0180 1      UBE_XFER_SIZE: VECTOR[4,WORD],
: 0181 1      UBE_STAT_ERR: BITVECTOR[4],
: 0182 1      UBE0_ISR,
: 0183 1      UBE1_ISR,
: 0184 1      UBE2_ISR,
: 0185 1      UBE3_ISR,
: 0186 1      UBI_UNDER_TEST,
: 0187 1      UNIBUS_SPACE,
: 0188 1      VALID_MAPS: BITVECTOR[496];
: 0189 1
: 0190 1      !
: 0191 1      ! MACROS:
: 0192 1      !
: 0193 1
: 0194 1      MACRO
: 0195 1      UBE_DB = (.UBE_BASE_ADDR[.N])<0,16>x,      ! UBE[N] Data buffer
: 0196 1      UBE_CC = (.UBE_BASE_ADDR[.N] + 2)<0,16>x, ! UBE[N] Cycle Count

```

2
)
ZZ-ECCBA-1.4 01-04
UBI SUBROUTINES
01-04

G 7

6-Sep-1985

Fiche 1 Frame G7

Sequence 84

6-Sep-1985 17:18:07

VAX-11 Bliss-32 V4.1-746

Page 5

6-Sep-1985 17:14:45

[LEMIEUX.ECCBA.RELEASE]ECCBA1.B32;3

(2)

: 0197 1 UBE_BA = (.UBE_BASE_ADDR[.N] + 4)<0,16>x, ! UBE[N] Bus address
: 0198 1 UBE_CR1 = (.UBE_BASE_ADDR[.N] + 6)<0,16>x, ! UBE[N] Control register 1
: 0199 1 UBE_CE = (.UBE_BASE_ADDR[.N] + 8)<0,16>x, ! UBE[N] Clear error
: 0200 1 UBE_SG = (.UBE_BASE_ADDR[.N] + 12)<0,16>x, ! UBE[N] Simultaneous go
: 0201 1 UBE_CR2 = (.UBE_BASE_ADDR[.N] + 14)<0,16>x, ! UBE[N] Control register 2

; 0202 1 GLOBAL ROUTINE PRINT_CSR_ERR (D1,D2,D3,D4,CSR_NUM,EXP_DATA,REC_DATA) :NOVALUE = ! M8

; 0203 1
 ; 0204 1 !**
 ; 0205 1 ! FUNCTIONAL DESCRIPTION:

; 0206 1 !
 ; 0207 1 ! This routine is invoked as a result of a Control and Status Register
 ; 0208 1 ! (CSR) error. It prints a header identifying the error as a CSR error,
 ; 0209 1 ! along with the following information:

; 0210 1 !
 ; 0211 1 ! CSR number
 ; 0212 1 ! Longword contents of the CSR
 ; 0213 1 ! Expected CSR contents and mnemonics
 ; 0214 1 ! Received CSR contents and mnemonics

; 0215 1 !
 ; 0216 1 ! FORMAL PARAMETERS:

; 0217 1 !
 ; 0218 1 ! CSR_NUM - CSR Number (0, 1, 2, or 3)
 ; 0219 1 ! EXP_DATA - expected contents of CSR
 ; 0220 1 ! REC_DATA - actual contents of CSR

; 0221 1 !
 ; 0222 1 ! IMPLICIT INPUTS: NONE

; 0223 1 !
 ; 0224 1 ! IMPLICIT OUTPUTS: NONE

; 0225 1 !
 ; 0226 1 ! COMPLETION CODES: NONE

; 0227 1 !
 ; 0228 1 ! SIDE EFFECTS:

; 0229 1 !
 ; 0230 1 ! A message is printed on the console.

; 0231 1 !--

; 0232 1
 ; 0233 2 BEGIN

; 0234 2
 ; 0235 2 LOCAL
 ; 0236 2 MNEMONICS:BLOCK(512,BYTE),
 ; 0237 2 CSR;
 ; 0238 2
 ; 0239 2 CSR = .REC_DATA AND CSR_MASK;

! A8

! A8

; 0240 2
 ; 0241 2 CODECOMMENT

; 0242 2
 ; 0243 2 'Print the error header and the CSR number.'
 ; 0244 2

; 0245 3 BEGIN
 ; 0246 3 \$DS_PRINTB(FMT_CSR_ERR1,.CSR_NUM);
 ; 0247 3 ! \$DS_PRINTX(FMT_CSR_ERR2,.CSR_NUM,.REC_DATA);
 ; 0248 2 END;

! D8

; 0249 2
 ; 0250 2 CODECOMMENT

; 0251 2
 ; 0252 2 'Convert each set bit into the bit mnemonics, and then print them.'
 ; 0253 2

; 0254 3 BEGIN
 ; P 0255 3 \$DS_CVTREG(MSB=31,DATA=.REC_DATA,MNEADR=CSR_MNEM,
 ; 0256 3 STRBUF=MNEMONICS,MAXLEN=512);

```

: 0257 3 $DS_PRINTX(FMT_BODY,EXP_MSG,.EXP_DATA,MNEMONICS); ! M8
: P 0258 3 $DS_CVTREG(MSB=31,DATA=.EXP_DATA,MNEADR=CSR_MNEM, ! A8
: 0259 3 STRBUF=MNEMONICS,MAXLEN=512); ! A8
: 0260 3 $DS_PRINTX(FMT_BODY,REC_MSG,.CSR,MNEMONICS); ! A8
: 0261 2 END;
: 0262 2
: 0263 1 END;

```

.TITLE UBI SUBROUTINES
 .IDENT \01-04\

```

DS$AL_APTMAIL= 65024
DS$GL_FLAGS= 65024
DS$GL_APTCOM= 65028
DS$GL_PASSES= 65032
DS$GL_UNITS= 65036
DS$GL_SECTNO= 65040
DS$GL_SID= 65044
DS$GL_MSGTYP= 65088
DS$GL_ERRNO= 65092
DS$GL_EVENT= 65096
DS$GL_SUBTNO= 65100
DS$GL_TESTNO= 65104
DS$GL_PASSNO= 65108
DS$GL_DEVLEN= 65112
DS$GL_DEVNAM= 65116
DS$GL_MSGPTR= 65128
.EXTRN CODEWORDS, CODEVECTORS
.EXTRN CSR_MNEM, DECODED_WORDS
.EXTRN EVENT_FLAG, EXP_MSG
.EXTRN EXP_REC_XOR, FMT_BODY
.EXTRN FMT_CSR_ERR1, FMT_CSR_ERR2
.EXTRN FMT_DCMP_ERR1, FMT_DCMP_ERR1A
.EXTRN FMT_DCMP_ERR2, FMT_MAP_GEN
.EXTRN FMT_NOTHING, FREE_MAP
.EXTRN LUN, NOISY_WORDS
.EXTRN NUM_MAPS, NUM_DATA_ERR
.EXTRN REC_MSG, UBE_BASE_ADDR
.EXTRN UBE_BUF_ADDR, UBE_MODE
.EXTRN UBE_MAP_NO, UBE_DP_NO
.EXTRN UBE_BUF_SIZE, UBE_PAGE_CNT
.EXTRN UBE_ERR_BUF, UBE_EXP_DATA
.EXTRN UBE_INIT_DATA, UBE_INT_OCC
.EXTRN UBE_XFER_ERR, UBE_XFER_SIZE
.EXTRN UBE_STAT_ERR, UBE0_ISR
.EXTRN UBE1_ISR, UBE2_ISR
.EXTRN UBE3_ISR, UBI_UNDER_TEST
.EXTRN UNIBUS_SPACE, VALID_MAPS
.EXTRN DS$PRINTB, DS$CVTREG
.EXTRN DS$PRINTX

```

.PSECT CODE,NOWRT,2

001C 00000

.ENTRY PRINT_CSR_ERR, Save R2,R3,R4

: 0202

54	00000000G	9F	9E	00002	MOVAB	a#DS\$PRINTX, R4	:
53	00000000G	9F	9E	00009	MOVAB	a#DS\$CVTREG, R3	:
5E	FE00	CE	9E	00010	MOVAB	-512(SP), SP	:
52	1C	AC	1FFFFFFE	8F	CB	00015	: 0239
							:
						; Print the error header and the CSR number.	:
							:
		14	AC	DD	0001E	PUSHL	CSR_NUM
		0000G	CF	9F	00021	PUSHAB	FMT_CSR_ERR1
00000000G	9F	02	FB	00025	CALLS	#2, a#DS\$PRINTB	: 0246
							:
						; Convert each set bit into the bit mnemonics, and then print t hem.	:
							:
			7E	7C	0002C	CLRQ	-(SP)
			7E	7C	0002E	CLRQ	-(SP)
			7E	7C	00030	CLRQ	-(SP)
7E	0200	8F	3C	00032	MOVZWL	#512, -(SP)	:
	1C	AE	9F	00037	PUSHAB	MNEMONICS	:
	0000G	CF	9F	0003A	PUSHAB	CSR_MNEM	:
	1C	AC	DD	0003E	PUSHL	REC_DATA	:
		1F	DD	00041	PUSHL	#31	:
63		0B	FB	00043	CALLS	#11, DS\$CVTREG	:
		5E	DD	00046	PUSHL	SP	: 0257
	18	AC	DD	00048	PUSHL	EXP_DATA	:
	0000G	CF	9F	0004B	PUSHAB	EXP_MSG	:
	0000G	CF	9F	0004F	PUSHAB	FMT_BODY	:
64		04	FB	00053	CALLS	#4, DS\$PRINTX	:
		7E	7C	00056	CLRQ	-(SP)	: 0259
		7E	7C	00058	CLRQ	-(SP)	:
		7E	7C	0005A	CLRQ	-(SP)	:
7E	0200	8F	3C	0005C	MOVZWL	#512, -(SP)	:
	1C	AE	9F	00061	PUSHAB	MNEMONICS	:
	0000G	CF	9F	00064	PUSHAB	CSR_MNEM	:
	18	AC	DD	00068	PUSHL	EXP_DATA	:
		1F	DD	0006B	PUSHL	#31	:
63		0B	FB	0006D	CALLS	#11, DS\$CVTREG	:
	4004	8F	BB	00070	PUSHR	#^M<R2, SP>	: 0260
	0000G	CF	9F	00074	PUSHAB	REC_MSG	:
	0000G	CF	9F	00078	PUSHAB	FMT_BODY	:
64		04	FB	0007C	CALLS	#4, DS\$PRINTX	:
		04	0007F		RET		: 0263

; Routine Size: 128 bytes, Routine Base: CODE + 0000


```

: 0264 1 GLOBAL ROUTINE PRINT_DCOMP_ERR (A,B,C,D,UBE_NUM) :NOVALUE =
: 0265 1
: 0266 1 !**
: 0267 1 ! FUNCTIONAL DESCRIPTION:
: 0268 1 !
: 0269 1 !     This routine is invoked as a result of a data compare error. It
: 0270 1 !     prints a header identifying the error as a data compare error, then
: 0271 1 !     prints the following information:
: 0272 1 !
: 0273 1 !         Transfer size in kilobytes
: 0274 1 !         Number of bad words
: 0275 1 !         Address of word in error
: 0276 1 !         Expected data
: 0277 1 !         Received data
: 0278 1 !         XOR of expected and received data
: 0279 1 !
: 0280 1 ! FORMAL PARAMETERS:
: 0281 1 !
: 0282 1 !     UBE_NUM - UBE number
: 0283 1 !
: 0284 1 ! IMPLICIT INPUTS:
: 0285 1 !
: 0286 1 !     UBE_XFER_SIZE[.UBE_NUM] - size of transfer in words
: 0287 1 !     NUM_DATA_ERR[.UBE_NUM] - number of bad words
: 0288 1 !     UBE_BUF_ADDR[.UBE_NUM] - address of memory data buffer
: 0289 1 !     UBE_EXP_DATA[.UBE_NUM] - expected data
: 0290 1 !     UBE_INIT_DATA[.UBE_NUM] - Initial data in buffer
: 0291 1 !     UBE_MODE[.UBE_NUM] - Byte offset flag
: 0292 1 !     UBE_DP_NO[N] - Data path number assigned to the UBE (-1 if rotating)
: 0293 1 !
: 0294 1 ! IMPLICIT OUTPUTS: NONE
: 0295 1 !
: 0296 1 ! COMPLETION CODES: NONE
: 0297 1 !
: 0298 1 ! SIDE EFFECTS:
: 0299 1 !
: 0300 1 !     A message is printed on the console.
: 0301 1 ! --
: 0302 1
: 0303 2 BEGIN
: 0304 2
: 0305 2 LOCAL
: 0306 2     XFER_SIZE,
: 0307 2     NUM_ERR,
: 0308 2     BUF_ADDR,
: 0309 2     WORD_NUM,
: 0310 2     WORD_ADDR,
: 0311 2     EXP_DATA: WORD,
: 0312 2     REC_DATA: WORD,
: 0313 2     XOR_DATA: WORD,
: 0314 2     DP_NUMB: BYTE;
: 0315 2
: 0316 2 BIND
: 0317 2     IO_BUF = .(UBE_BUF_ADDR[.UBE_NUM]);
: 0318 2

```

```

: 0319 2  MAP
: 0320 2      IO_BUF: VECTOR[65536,WORD];
: 0321 2
: 0322 2  XFER_SIZE = .UBE_XFER_SIZE[.UBE_NUM]*2;
: 0323 2  NUM_ERR = .NUM_DATA_ERR[.UBE_NUM];
: 0324 2  BUF_ADDR = .UBE_BUF_ADDR[.UBE_NUM];
: 0325 2
: 0326 2  IF .UBE_DP_NO[.UBE_NUM] NEQ &X'FF'
: 0327 2  THEN DP_NUMB = .UBE_DP_NO[.UBE_NUM]
: 0328 2  ELSE DP_NUMB = 4;
: P 0329 2  $DS_PRINTX(FMT_DCMP_ERR1,.XFER_SIZE,IO_BUF[0],.NUM_ERR,.UBE_NUM,
: 0330 2      .DP_NUMB,.UBE_MODE[.UBE_NUM]);      ! M10
: 0331 2  $DS_PRINTX(FMT_DCMP_ERR1A);
: 0332 2
: 0333 2  IF .NUM_ERR GTR 64                        ! A8
: 0334 2  THEN NUM_ERR = 64;                        ! A8
: 0335 2
: 0336 2
: 0337 2  INCR N FROM 0 TO .NUM_ERR - 1 DO
: 0338 3      BEGIN
: 0339 3          WORD_NUM = .UBE_ERR_BUF[.UBE_NUM,.N];      ! M8
: 0340 3
: 0341 3          IF .UBE_MODE[.UBE_NUM] NEQ 0                ! A10
: 0342 3          THEN
: 0343 4              BEGIN
: 0344 4                  EXP_DATA<0,8> = .(UBE_EXP_DATA[.UBE_NUM])<8,8>;      ! A10
: 0345 4                  EXP_DATA<8,8> = .(UBE_EXP_DATA[.UBE_NUM])<0,8>;
: 0346 4                  IF .WORD_NUM EQL 0                    ! A10
: 0347 4                  THEN EXP_DATA<0,8> = .(UBE_INIT_DATA[.UBE_NUM])<0,8> ! A10
: 0348 4                  ELSE                                  ! A10
: 0349 5                      BEGIN                                ! A10
: 0350 5                          IF .WORD_NUM EQL .UBE_XFER_SIZE[.UBE_NUM] ! A10
: 0351 5                          THEN EXP_DATA<8,8> = .(UBE_INIT_DATA[.UBE_NUM])<8,8>; ! A10
: 0352 4                      END;                                ! A10
: 0353 4                  END                                    ! A10
: 0354 3          ELSE EXP_DATA = .UBE_EXP_DATA[.UBE_NUM];      ! A10
: 0355 3          WORD_ADDR = .BUF_ADDR + .WORD_NUM*2;
: 0356 3          REC_DATA = .IO_BUF[.WORD_NUM];
: 0357 3          XOR_DATA = .EXP_DATA XOR .REC_DATA;
: 0358 3          $DS_PRINTX(FMT_DCMP_ERR2,.WORD_ADDR,.EXP_DATA,.REC_DATA,.XOR_DATA);
: 0359 2          END;
: 0360 2
: 0361 1  END;

```

```

OFFC 00000
5E          04 C2 00002
53          14 AC D0 00005
5A          0000GCF43 D0 00009
51          0000GCF43 3C 0000F
51          02 C4 00015

```

```

.ENTRY PRINT_DCMP_ERR, Save R2,R3,R4,R5,R6,R7,R8,- ; 0264
R9,R10,R11
:
:
:
:
: 0317
:
: 0322
:
:
:
:

```

		58	0000GCF43	D0	00018	MOVL	NUM_DATA_ERR[R3], NUM_ERR	; 0323
			5A	DD	0001E	PUSHL	R10	; 0324
	FF	8F	0000GCF43	91	00020	CMPB	UBE_DP_NO[R3], #255	; 0326
			08	13	00027	BEQL	1\$	; 0327
		50	0000GCF43	90	00029	MOVB	UBE_DP_NO[R3], DP_NUMB	; 0328
			03	11	0002F	BRB	2\$	; 0330
		50		04	90	00031	1\$: MOVB	#4, DP_NUMB
		7E	0000GCF43	9A	00034	2\$: MOVZBL	UBE_MODE[R3], -(SP)	; 0331
		7E		50	9A	0003A	MOVZBL	DP_NUMB, -(SP)
				53	DD	0003D	PUSHL	R3
				58	DD	0003F	PUSHL	NUM_ERR
			0402	8F	BB	00041	PUSHR	#^M<R1,R10>
			0000G	CF	9F	00045	PUSHAB	FMT_DCMP_ERR1
	000000000G	9F		07	FB	00049	CALLS	#7, a#DS\$PRINTX
			0000G	CF	9F	00050	PUSHAB	FMT_DCMP_ERR1A
	000000000G	9F		01	FB	00054	CALLS	#1, a#DS\$PRINTX
	000000040	8F		58	D1	0005B	CMPL	NUM_ERR, #64
				04	15	00062	BLEQ	3\$
		58	40	8F	9A	00064	MOVZBL	#64, NUM_ERR
59		53		06	78	00068	3\$: ASHL	#6, R3, R9
		56	0000GCF43	3E	0006C	MOVAV	UBE_EXP_DATA[R3], R6	; 0344
		52		01	CE	00072	MNEGL	#1, N
				66	11	00075	BRB	8\$
50		52		59	C1	00077	4\$: ADDL3	R9, N, R0
		54	0000GCF40	D0	0007B	MOVL	UBE_ERR_BUF[R0], WORD_NUM	; 0341
			0000GCF43	95	00081	TSTB	UBE_MODE[R3]	; 0344
				2D	13	00086	BEQL	6\$
55		55	01	A6	90	00088	MOVB	1(R6), EXP_DATA
	08	08		66	F0	0008C	INSV	(R6), #8, #8, EXP_DATA
				54	D5	00091	TSTL	WORD_NUM
				08	12	00093	BNEQ	5\$
		55	0000GCF43	33	00095	CVTWB	UBE_INIT_DATA[R3], EXP_DATA	; 0347
				1B	11	0009B	BRB	7\$
			0000GCF43	3F	0009D	5\$: PUSHAW	UBE_XFER_SIZE[R3]	; 0350
54		10		00	ED	000A2	CMPZV	#0, #16, a(SP)+, WORD_NUM
				0F	12	000A7	BNEQ	7\$
			0000GCF43	3F	000A9	PUSHAW	UBE_INIT_DATA+1[R3]	; 0351
55		08		9E	F0	000AE	INSV	a(SP)+, #8, #8, EXP_DATA
				03	11	000B3	BRB	7\$
		55		66	B0	000B5	6\$: MOVW	(R6), EXP_DATA
	04	AE	00	BE44	3E	000B8	7\$: MOVAV	aBUF_ADDR[WORD_NUM], WORD_ADDR
		57		6A44	B0	000BE	MOVW	(R10)[WORD_NUM], REC_DATA
5B		55		57	AD	000C2	XORW3	REC_DATA, EXP_DATA, XOR_DATA
		7E		5B	3C	000C6	MOVZWL	XOR_DATA, -(SP)
		7E		57	3C	000C9	MOVZWL	REC_DATA, -(SP)
		7E		55	3C	000CC	MOVZWL	EXP_DATA, -(SP)
			10	AE	DD	000CF	PUSHL	WORD_ADDR
			0000G	CF	9F	000D2	PUSHAB	FMT_DCMP_ERR2
	000000000G	9F		05	FB	000D6	CALLS	#5, a#DS\$PRINTX
96		52		58	F2	000DD	8\$: AOBLS	NUM_ERR, N, 4\$
				04	000E1	RET		; 0361

; Routine Size: 226 bytes, Routine Base: CODE + 0080


```

; 0362 1 GLOBAL ROUTINE PRINT_MAP_ERR(D1,D2,D3,D4,MAP_NUM,MAP_ADDRESS,MSG_ADDRESS,EXP_DATA,REC_DATA) :NOVALUE =
; 0363 1
; 0364 1 !+
; 0365 1 ! FUNCTIONAL DESCRIPTION:
; 0366 1 !
; 0367 1 ! This routine is invoked via a PRINT LINK to type a map error
; 0368 1 ! message, the map numebr, address, and expected and received data.
; 0369 1 !
; 0370 1 ! FORMAL PARAMETERS:
; 0371 1 !
; 0372 1 ! MAP_NUM - Map number that failed.
; 0373 1 ! MAP_ADDRESS - Address of map that failed.
; 0374 1 ! MSG_ADDRESS - Address of error message.
; 0375 1 ! EXP_DATA - Expected data
; 0376 1 ! REC_DATA - Received data
; 0377 1 !
; 0378 1 !--
; 0379 1
; 0380 2 BEGIN
; 0381 2
; 0382 2 LOCAL
; 0383 2 EXPECTED_DATA,
; 0384 2 RECEIVED_DATA;
; 0385 2
; 0386 2 EXPECTED_DATA = .EXP_DATA AND MAP_MASK;
; 0387 2 RECEIVED_DATA = .REC_DATA AND MAP_MASK;
; 0388 2
; 0389 2 CODECOMMENT
; 0390 2 '
; 0391 2 'Print the map number and address.',
; 0392 2 '
; 0393 3 BEGIN
; 0394 3 $DS_PRINTB(.MSG_ADDRESS);
; 0395 3 $DS_PRINTB(FMT_MAP_GEN,.MAP_NUM,.MAP_ADDRESS);
; 0396 2 END;
; 0397 2
; 0398 2 CODECOMMENT
; 0399 2 '
; 0400 2 'Print the expected, received, and XOR data.',
; 0401 2 '
; 0402 3 BEGIN
; P 0403 3 $DS_PRINTB(EXP_REC_XOR,.EXPECTED_DATA,.RECEIVED_DATA,
; 0404 3 (.EXPECTED_DATA XOR .RECEIVED_DATA));
; 0405 2 END;
; 0406 1 END;

```

				001C 00000	.ENTRY PRINT_MAP_ERR, Save R2,R3,R4	: 0362
		54 00000000G	9F 9E 00002	MOVAB	a#DS\$PRINTB, R4	:
53	20	AC 7D9F8000	8F CB 00009	BICL3	#2107604992, EXP_DATA, EXPECTED_DATA	: 0386
52	24	AC 7D9F8000	8F CB 00012	BICL3	#2107604992, REC_DATA, RECEIVED_DATA	: 0387

; Print the map number and address.

		1C	AC	DD	0001B	PUSHL	MSG_ADDRESS		; 0394
64			01	FB	0001E	CALLS	#1, DS\$PRINTB		; 0395
7E		14	AC	7D	00021	MOVQ	MAP_NUM, -(SP)		
	0000G		CF	9F	00025	PUSHAB	FMT_MAP_GEN		
64			03	FB	00029	CALLS	#3, DS\$PRINTB		

; Print the expected, received, and XOR data.

7E		53		52	CD	0002C	XORL3	RECEIVED_DATA, EXPECTED_DATA, -(SP)	; 0404
				52	DD	00030	PUSHL	RECEIVED_DATA	
				53	DD	00032	PUSHL	EXPECTED_DATA	
	0000G			CF	9F	00034	PUSHAB	EXP_REC_XOR	
64			04	FB	00038	CALLS	#4, DS\$PRINTB		
				04	0003B	RET			; 0406

; Routine Size: 60 bytes, Routine Base: CODE + 0162

```

: 0407 1 GLOBAL ROUTINE PRINT_GENERAL(D1,D2,D3,D4,MSG_ADDR,PARA_CNT,P1,P2,P3,P4) : NOVALUE =
: 0408 1 !++
: 0409 1 ! FUNCTIONAL DESCRIPTION:
: 0410 1 !
: 0411 1 !     This routine is called via a print link. It is used to print
: 0412 1 !     a basic error message.
: 0413 1 !
: 0414 1 ! FORMAL PARAMETERS:
: 0415 1 !
: 0416 1 !     MSG_ADDR      - Address of ascii error message
: 0417 1 !     PARA_CNT     - Number of parameters
: 0418 1 !     P1 - P4      - Parameters that are part of error report
: 0419 1 !
: 0420 1 ! --
: 0421 1
: 0422 2 BEGIN
: 0423 2 CASE PARA_CNT FROM 0 TO 4 OF
: 0424 2 SET
: 0425 2 [0]: $DS_PRINTB(.MSG_ADDR);
: 0426 2 [1]: $DS_PRINTB(.MSG_ADDR,.P1);
: 0427 2 [2]: $DS_PRINTB(.MSG_ADDR,.P1,.P2);
: 0428 2 [3]: $DS_PRINTB(.MSG_ADDR,.P1,.P2,.P3);
: 0429 2 [4]: $DS_PRINTB(.MSG_ADDR,.P1,.P2,.P3,.P4);
: 0430 2 TES;
: 0431 1 END;

```

				000C 00000	.ENTRY PRINT_GENERAL, Save R2,R3	: 0407
		53 00000000G	9F 9E 00002	MOVAB	a#DS\$PRINTB, R3	: 0425
		52 14 AC D0 00009		MOVL	MSG_ADDR, R2	: 0423
		00 18 AC CF 0000D		CASEL	PARA_CNT, #0, #4	:
0023	04	0010 000A 00012 1\$:		.WORD	2\$-1\$,-	:
		0030 0001A			3\$-1\$,-	:
					4\$-1\$,-	:
					5\$-1\$,-	:
					6\$-1\$	:
		52 DD 0001C 2\$:		PUSHL	R2	: 0425
		63 01 FB 0001E		CALLS	#1, DS\$PRINTB	:
		04 00021		RET		:
		1C AC DD 00022 3\$:		PUSHL	P1	: 0426
		52 DD 00025		PUSHL	R2	:
		63 02 FB 00027		CALLS	#2, DS\$PRINTB	:
		04 0002A		RET		:
		7E 1C AC 7D 0002B 4\$:		MOVQ	P1, -(SP)	: 0427
		52 DD 0002F		PUSHL	R2	:
		63 03 FB 00031		CALLS	#3, DS\$PRINTB	:
		04 00034		RET		:
		7E 20 AC 7D 00035 5\$:		MOVQ	P2, -(SP)	: 0428
		1C AC DD 00039		PUSHL	P1	:
		52 DD 0003C		PUSHL	R2	:
		63 04 FB 0003E		CALLS	#4, DS\$PRINTB	:
		04 00041		RET		:

2
)
ZZ-ECCBA-1.4 01-04
UBI SUBROUTINES
01-04

D 8

6-Sep-1985

Fiche 1 Frame D8

Sequence 94

6-Sep-1985 17:18:07

VAX-11 Bliss-32 V4.1-746

Page 15

6-Sep-1985 17:14:45

[LEMIEUX.ECCBA.RELEASE]ECCBA1.B32;3

(6)

7E	24	AC	7D 00042	6\$:	MOVQ	P3, -(SP)	; 0429
7E	1C	AC	7D 00046		MOVQ	P1, -(SP)	:
		52	DD 0004A		PUSHL	R2	:
63		05	FB 0004C		CALLS	#5, DS\$PRINTB	:
		04	0004F		RET		; 0431

; Routine Size: 80 bytes, Routine Base: CODE + 019E

; 0432 1

```

: 0433 1 GLOBAL ROUTINE HANDLER(SIG,MECH,ENBL) = ! Condition handler.
: 0434 1
: 0435 1 !**
: 0436 1 ! FUNCTIONAL DESCRIPTION:
: 0437 1 !
: 0438 1 ! This routine is invoked by an exception. In general, a machine check
: 0439 1 ! is the only type of exception that should occur during execution of
: 0440 1 ! the program. Whenever an action by the diagnostic program may cause a
: 0441 1 ! machine check (such as the first time the program attempts to access a
: 0442 1 ! UBI register, or the normal execution of the Unibus sizer routine) the
: 0443 1 ! program will enable this routine. At all other times all exceptions
: 0444 1 ! will be handled by the Diagnostic Suprvisor.
: 0445 1 !
: 0446 1 ! This routine determines whether or not the exception which has invoked
: 0447 1 ! it is a machine check; if it was not, the handler returns to the
: 0448 1 ! Supervisor (that is, it resignals), and the Supervisor will handle the
: 0449 1 ! exception. If the invoking exception was a machine check, the handler
: 0450 1 ! unwinds the stack one frame so the the return will be from the the
: 0451 1 ! routine which caused the machine check.
: 0452 1 !
: 0453 1 ! FORMAL PARAMETERS:
: 0454 1 !
: 0455 1 ! SIG - contains the address of a signal vector, which is a counted
: 0456 1 ! vector used to access the type of condition.
: 0457 1 !
: 0458 1 ! MECH - contains the address of a mechanism vector, which is a counted
: 0459 1 ! vector used to access the saved return value.
: 0460 1 !
: 0461 1 ! ENBL - contains the address of an enable vector, which is a counted
: 0462 1 ! vector that contains the values of the enable-actual
: 0463 1 ! parameters of the ENABLE declaration of the establisher
: 0464 1 ! routine.
: 0465 1 !
: 0466 1 ! IMPLICIT INPUTS: NONE
: 0467 1 !
: 0468 1 ! IMPLICIT OUTPUTS: NONE
: 0469 1 !
: 0470 1 ! ROUTINE VALUE:
: 0471 1 !
: 0472 1 ! 0
: 0473 1 !
: 0474 1 ! SIDE EFFECTS: NONE
: 0475 1 !
: 0476 1 ! --
: 0477 1
: 0478 2 BEGIN
: 0479 2
: 0480 2 MAP
: 0481 2 SIG: REF VECTOR, ! Signal vector
: 0482 2 MECH: REF VECTOR, ! Mechanism vector
: 0483 2 ENBL: REF VECTOR; ! Enable vector
: 0484 2
: 0485 2 BIND
: 0486 2 COND = SIG[1], ! Condition identifier
: 0487 2 RETURN_VALUE = MECH[3]; ! Value of unwound return

```

```

; 0488 2
; 0489 2 CODECOMMENT
; 0490 2
; 0491 2 'Determine whether or not the exception was a machine check. If it was,',
; 0492 2 'then unwind the stack one frame and cause the value of the saved return',
; 0493 2 'value to be zero. Otherwise, resignal (return to the Supervisor',
; 0494 2 'condition handler facility with the same condition value).',
; 0495 2
; 0496 3 BEGIN
; 0497 3 IF .COND EQL DS$_MCHK ! If the condition was a machine check
; 0498 3 THEN ! Then do the following:
; 0499 4 BEGIN ! Set the routine value of the
; 0500 4 RETURN_VALUE = 0; ! establisher routine to 0 and
; 0501 4 SETUNWIND(); ! unwind the stack one call frame
; 0502 4 END
; 0503 3 ELSE ! Otherwise,
; 0504 3 RETURN SS$_RESIGNAL; ! resignal the Supervisor
; 0505 2 END;
; 0506 2
; 0507 2 RETURN SS$_NORMAL; ! Normal completion
; 0508 2
; 0509 1 END;

```

51	04	AC	0000 00000	.ENTRY HANDLER, Save nothing	; 0433
50	08	AC	04 C1 00002	ADDL3 #4, SIG, R1	; 0486
			0C C1 00007	ADDL3 #12, MECH, R0	; 0487
; Determine whether or not the exception was a machine check.					
; If it was,					
; then unwind the stack one frame and cause the value of the saved return					
; value to be zero. Otherwise, resignal (return to the Supervisor					
; condition handler facility with the same condition value).					
00660088	8F	61	D1 0000C	CMPL (R1), #6684808	; 0497
		0D	12 00013	BNEQ 1\$	; 0500
		60	D4 00015	CLRL (R0)	; 0501
		7E	7C 00017	CLRQ -(SP)	; 0497
00000000G	00	02	FB 00019	CALLS #2, SYS\$UNWIND	; 0504
	50	06	11 00020	BRB 2\$	; 0507
		8F	3C 00022 1\$:	MOVZWL #2328, R0	; 0509
	50	04	00027	RET	
		01	D0 00028 2\$:	MOVL #1, R0	
		04	0002B	RET	

; Routine Size: 44 bytes, Routine Base: CODE + 01EE


```

; 0510 1 GLOBAL ROUTINE UBE_INTERRUPT (UBE_NUM): NOVALUE =
; 0511 1
; 0512 1 !++
; 0513 1 ! FUNCTIONAL DESCRIPTION:
; 0514 1 !
; 0515 1 !     This routine is called by a UBE interrupt service routine. A UBE
; 0516 1 !     will normally interrupt at the completion of a block transfer, which
; 0517 1 !     is signalled by the overflow of the UBE cycle count register. An
; 0518 1 !     interrupt may also occur if the error bit is set in UBE CR1. This
; 0519 1 !     routine raises the IPL to prevent any further UBE interrupts at BR7 or
; 0520 1 !     lower, determines whether or not an error has occurred, and if not,
; 0521 1 !     calls the transfer analysis routine.
; 0522 1 !
; 0523 1 ! FORMAL PARAMETERS:
; 0524 1 !
; 0525 1 !     UBE_NUM - indicates which UBE requesting service
; 0526 1 !
; 0527 1 ! IMPLICIT INPUTS: NONE
; 0528 1 !
; 0529 1 ! IMPLICIT OUTPUTS:
; 0530 1 !
; 0531 1 !     UBE_STAT_ERR[N] - a flag indicating a status error occurred
; 0532 1 !     UBE_INT_OCC[N] - a flag indicating an interrupt occurred.
; 0533 1 !
; 0534 1 ! COMPLETION CODES: NONE
; 0535 1 !
; 0536 1 ! SIDE EFFECTS: NONE
; 0537 1 !
; 0538 1 !--
; 0539 1
; 0540 2 BEGIN
; 0541 2
; 0542 2 CODECOMMENT
; 0543 2 ''
; 0544 2 'Raise IPL to 17 (hexadecimal), disallowing other UBE bus requests.',
; 0545 2 ''
; 0546 3     BEGIN
; 0547 3     $DS_SETIPL(LEVEL=%X'17');
; 0548 2     END;
; 0549 2
; 0550 2 CODECOMMENT
; 0551 2 ''
; 0552 2 'Set the Interrupt Occurred Flag',
; 0553 2 ''
; 0554 3     BEGIN
; 0555 3     UBE_INT_OCC[.UBE_NUM] = 1;
; 0556 2     END;
; 0557 2
; 0558 2 CODECOMMENT
; 0559 2 ''
; 0560 2 'Read UBE Control Register 1 (UBE CR1). If the error bit is clear,',
; 0561 2 'then call the transfer analysis routine. Otherwise, set a bit',
; 0562 2 'indicating UBE has a status error.',
; 0563 2 ''
; 0564 3     BEGIN

```

```

! A8
! A8
! A8
! A8
! A8
! A8
! A8

```

```

: 0565 3      REGISTER
: 0566 3          CR1,
: 0567 3          N;
: 0568 3
: 0569 3      N = .UBE_NUM;
: 0570 3      CR1 = .UBE_CR1;
: 0571 3
: 0572 3      IF (.CR1 AND ERROR) NEQ 0
: 0573 3      THEN
: 0574 3          UBE_STAT_ERR[.N] = 1
: 0575 3      ELSE
: 0576 3          XFER_ANALYSIS(.N);
: 0577 3
: 0578 2      END;
: 0579 2
: 0580 2      !CODECOMMENT
: 0581 2      !'',
: 0582 2      !'Lower IPL to 13 (hexadecimal), allowing other UBE bus requests.', ! D11
: 0583 2      !'',
: 0584 2      !      BEGIN
: 0585 2      !      $DS_SETIPL(LEVEL=XX'13');
: 0586 2      !      END;
: 0587 2
: 0588 1      END;

```

```

                                .EXTRN DS$SETIPL
                                .ENTRY UBE_INTERRUPT, Save nothing                ; 0510
                                ; Raise IPL to 17 (hexadecimal), disallowing other UBE bus requ
                                ; ests.
                                ;
00000000G   9F               17 DD 00002          PUSHL    #23                      ; 0547
                                01 FB 00004          CALLS    #1, a#DS$SETIPL              ;
                                ; Set the Interrupt Occurred Flag
                                ;
                                04 BC40           0000G CF 9E 0000B      MOVAB     UBE_INT_OCC, R0                  ; 0555
                                01 90 00010        MOVB     #1, aUBE_NUM[R0]            ;
                                ; Read UBE Control Register 1 (UBE CR1). If the error bit is c
                                ; lear,
                                ; then call the transfer analysis routine. Otherwise, set a bi
                                ; t
                                ; indicating UBE has a status error.
                                ;
                                51             04 AC D0 00015          MOVL     UBE_NUM, N                        ; 0569
                                50             0000GCF41 D0 00019       MOVL     UBE_BASE_ADDR[N], R0              ; 0570
                                50             06 A0 3C 0001F         MOVZWL   6(R0), CR1                       ;
                                50             50 B5 00023          TSTW     CR1                          ; 0572
                                07             18 00025          BGEQ    1$                               ;
08             0000G   CF           51 E2 00027        BBSS    N, UBE_STAT_ERR, 2$              ; 0574
                                04 0002D          RET                               ;

```

ZZ-ECCBA-1.4 01-04
UBI SUBROUTINES
01-04

I 8
6-Sep-1985 Fiche 1 Frame I8 Sequence 99
6-Sep-1985 17:18:07 VAX-11 Bliss-32 V4.1-746 Page 20
6-Sep-1985 17:14:45 [LEMIEUX.ECCBA.RELEASE]ECCBA1.B32;3 (8)

0000V	CF	51	DD	0002E	1\$:	PUSHL	N		; 0576
		01	FB	00030		CALLS	#1,	XFER_ANALYSIS	; 0588
		04		00035	2\$:	RET			

; Routine Size: 54 bytes, Routine Base: CODE + 021A


```

: 0589 1 GLOBAL ROUTINE XFER_ANALYSIS (UBE_NUM) :NOVALUE = ! Transfer analysis routine.
: 0590 1
: 0591 1 !**
: 0592 1 ! FUNCTIONAL DESCRIPTION:
: 0593 1
: 0594 1 ! This routine is called by a UBE interrupt service routine. It
: 0595 1 ! analyzes a DATO transaction upon its successful completion (that is,
: 0596 1 ! with no error status).
: 0597 1
: 0598 1 ! FORMAL PARAMETERS:
: 0599 1
: 0600 1 ! UBE_NUM - number of UBE completing the transfer
: 0601 1
: 0602 1 ! IMPLICIT INPUTS:
: 0603 1
: 0604 1 ! UBE_XFER_SIZE[N] - transfer size in cycles
: 0605 1 ! UBE_BUF_ADDR[N] - base address of buffer
: 0606 1 ! UBE_EXP_DATA[N] - expected data in buffer
: 0607 1 ! UBE_INIT_DATA[N] - Initial data in buffer
: 0608 1 ! UBE_MODE[N] - Non zero means byte offset mode
: 0609 1
: 0610 1 ! IMPLICIT OUTPUTS:
: 0611 1
: 0612 1 ! UBE_XFER_ERR[N] - 0 if no error detected in a DATO transaction,
: 0613 1 ! 1 if error detected
: 0614 1 ! UBE_ERR_BUF[N] - a list of pointers to actual data in the buffer
: 0615 1
: 0616 1 ! COMPLETION CODES: NONE
: 0617 1
: 0618 1 ! SIDE EFFECTS: NONE
: 0619 1
: 0620 1 !--
: 0621 1
: 0622 2 BEGIN
: 0623 2
: 0624 2 REGISTER
: 0625 2 ! ERR_CTR, ! An error counter max 64 ! M8
: 0626 2 ! TOTAL_ERRORS, ! Total number of errors ! A8
: 0627 2 ! EXP_DATA: WORD, ! Temp for expected data ! A10
: 0628 2 ! SIZE, ! Transfer size in words ! A10
: 0629 2 ! M, ! Word counter
: 0630 2 ! N; ! Indicates UBE number
: 0631 2
: 0632 2 CODECOMMENT
: 0633 2 ''
: 0634 2 'Set N = UBE_NUM, and clear the transfer error flag for UBE(N).',
: 0635 2 ''
: 0636 3 BEGIN
: 0637 3 N = .UBE_NUM; ! Set N to UBE number
: 0638 3 UBE_XFER_ERR[.N] = 0; ! Clear UBE_XFER_ERR flag
: 0639 2 END;
: 0640 2
: 0641 2 CODECOMMENT
: 0642 2 ''
: 0643 2 'Set an error counter to zero, and begin checking each word in the',

```

```

: 0644 2 'buffer against the expected data. Any error in this comparison will',
: 0645 2 'cause a pointer to the actual data in error to be stored in the buffer',
: 0646 2 'UBE_ERR_BUF[N] and also will cause the error counter to be incremented.',
: 0647 2 'Any error detected here will also cause the flag UBE_XFER_ERR[N] to be',
: 0648 2 'set.',
: 0649 2 '':
: 0650 3 BEGIN
: 0651 3 BIND
: 0652 3     IO_BUF = .(UBE_BUF_ADDR[.N]); ! Points to the IO buffer used in
: 0653 3                                     ! the transfer
: 0654 3 MAP
: 0655 3     IO_BUF:VECTOR[65536,WORD]; ! Gives the IO buffer the attributes
: 0656 3                                     ! of a vector made up of 64k words
: 0657 3 ERR_CTR = 0;
: 0658 3 TOTAL_ERRORS = 0; ! A8
: 0659 3 SIZE = .UBE_XFER_SIZE[.N] - 1; ! A10
: 0660 3 IF .UBE_MODE[.N] NEQ 0 ! A10
: 0661 3 THEN SIZE = .SIZE + 1; ! A10
: 0662 3 INCR M FROM 0 TO .SIZE DO ! M10
: 0663 4 BEGIN
: 0664 4     IF .UBE_MODE[.N] NEQ 0 ! A10
: 0665 4     THEN ! A10
: 0666 5 BEGIN ! A10
: 0667 5     EXP_DATA<0,8> = .(UBE_EXP_DATA[.N])<8,8>; ! A10
: 0668 5     EXP_DATA<8,8> = .(UBE_EXP_DATA[.N])<0,8>; ! A10
: 0669 5     IF .M EQL 0 ! A10
: 0670 5     THEN EXP_DATA<0,8> = .(UBE_INIT_DATA[.N])<0,8> ! A10
: 0671 5     ELSE ! A10
: 0672 6 BEGIN ! A10
: 0673 6     IF .M EQL .SIZE ! A10
: 0674 6     THEN EXP_DATA<8,8> = .(UBE_INIT_DATA[.N])<8,8>; ! A10
: 0675 5     END; ! A10
: 0676 5 END ! A10
: 0677 4 ELSE EXP_DATA = .UBE_EXP_DATA[.N]; ! A10
: 0678 4 IF .IO_BUF[M] NEQ .EXP_DATA ! M10
: 0679 4 THEN
: 0680 5 BEGIN
: 0681 5     IF .ERR_CTR LSS 64
: 0682 5     THEN
: 0683 6 BEGIN
: 0684 6     UBE_ERR_BUF[.N,.ERR_CTR] = .M; ! Save pointer to error ! M8
: 0685 6     ERR_CTR = .ERR_CTR + 1; ! Increment the error counter
: 0686 6     TOTAL_ERRORS = .TOTAL_ERRORS + 1; ! A8
: 0687 6     UBE_XFER_ERR[.N] = 1; ! Set the flag UBE_XFER_ERR
: 0688 6     END ! M8
: 0689 5 ELSE
: 0690 5     TOTAL_ERRORS = .TOTAL_ERRORS + 1; ! A8
: 0691 4 END;
: 0692 3 END;
: 0693 3 NUM_DATA_ERR[.N] = .TOTAL_ERRORS; ! Save total number of errors ! M8
: 0694 2 END;
: 0695 2
: 0696 1 END;

```

PC	Op	OpC	OpD	OpE	OpF	OpG	OpH	OpI	OpJ	OpK	OpL	OpM	OpN	OpO	OpP	OpQ	OpR	OpS	OpT	OpU	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

ZZ-ECCBA-1.4 01-04
UBI SUBROUTINES
01-04

M 8
6-Sep-1985 Fiche 1 Frame M8 Sequence 103
6-Sep-1985 17:18:07 VAX-11 Bliss-32 V4.1-746 Page 24
6-Sep-1985 17:14:45 [LEMIEUX.ECCBA.RELEASE]ECCBA1.B32;3 (9)

		50	D6	00078	INCL	ERR_CTR	; 0685
		51	D6	0007A	INCL	TOTAL_ERRORS	; 0686
04	0000G	54	E2	0007C	BBSS	N, UBE_XFER_ERR, 8\$	; 0687
		02	11	00082	BRB	8\$	; 0681
		51	D6	00084	INCL	TOTAL_ERRORS	; 0690
AA		53	F3	00086	AOBLEQ	SIZE, M, 3\$	; 0662
	56	51	D0	0008A	MOVL	TOTAL_ERRORS, NUM_DATA_ERR[N]	; 0693
	0000GCF44	04	00090	RET			; 0696

; Routine Size: 145 bytes, Routine Base: CODE + 0250

```

; 0697 1 GLOBAL ROUTINE PROBE_ADDRESS (PROBE_ADDR,LENGTH,INTERLOCK) = ! Probe address routine.
; 0698 1
; 0699 1 !++
; 0700 1 ! FUNCTIONAL DESCRIPTION:
; 0701 1 !
; 0702 1 ! This routine is invoked by the test modules whenever it is anticipated
; 0703 1 ! that an access of a particular location may yield a machine check.
; 0704 1 ! The routine enables the local exception handler, and then reads the
; 0705 1 ! desired address. If the address exists, then the routine returns
; 0706 1 ! normal status (routine value of 1). If the address is non-existent, a
; 0707 1 ! machine check is caused, and the exception handler is thus invoked.
; 0708 1 ! The exception handler will unwind the stack to cause the return to be
; 0709 1 ! 'from' this routine, and returns a value of 0, indicating that the
; 0710 1 ! address did not exist.
; 0711 1 !
; 0712 1 ! If the INTERLOCK parameter is non zero, the routine will perform
; 0713 1 ! an INTERLOCK read to the address (via an ADAWI instruction) and
; 0714 1 ! perform as described above.
; 0715 1 !
; 0716 1 ! FORMAL PARAMETERS:
; 0717 1 !
; 0718 1 ! PROBE_ADDR - address to probe
; 0719 1 ! LENGTH - length of location to probe: 1 if long, 0 if word
; 0720 1 ! INTERLOCK - Type of probe to perform: 0 if normal, 1 if INTERLOCK
; 0721 1 !
; 0722 1 ! IMPLICIT INPUTS: NONE
; 0723 1 !
; 0724 1 ! IMPLICIT OUTPUTS: NONE
; 0725 1 !
; 0726 1 ! ROUTINE VALUE:
; 0727 1 !
; 0728 1 ! SS$_NORMAL if address exists
; 0729 1 !
; 0730 1 ! SIDE EFFECTS: NONE
; 0731 1 ! --
; 0732 1 !
; 0733 2 BEGIN
; 0734 2 BUILTIN
; 0735 2 ADAWI;
; 0736 2 REGISTER
; 0737 2 PROBE_LONG,
; 0738 2 PROBE_WORD: WORD;
; 0739 2 LOCAL
; 0740 2 TEMP: WORD;
; 0741 2
; 0742 2 ENABLE HANDLER;
; 0743 2
; 0744 2 CODECOMMENT
; 0745 2 ''
; 0746 2 'Read the address contained in PROBE_ADDR. If it causes a machine check',
; 0747 2 'the condition handler routine (HANDLER) will be called by the Supervisor',
; 0748 2 'Condition Handler Facility (CHF). That routine will unwind the stack',
; 0749 2 'one frame so that the return will be from this routine.',
; 0750 2 ''
; 0751 3 BEGIN

```

```

; 0752 3
; 0753 3      IF .INTERLOCK EQL 0
; 0754 4      THEN BEGIN
; 0755 4          IF .LENGTH
; 0756 4          THEN PROBE_LONG = ..PROBE_ADDR
; 0757 4          ELSE PROBE_WORD = ..PROBE_ADDR;
; 0758 4          END
; 0759 3      ELSE
; 0760 4          BEGIN
; 0761 4              TEMP = 1;                ! A12 Initialize for ADAWI
; 0762 4              ADAWI(TEMP, ..PROBE_ADDR); ! M12
; 0763 3          END;
; 0764 2      END;
; 0765 2
; 0766 2      RETURN SS$_NORMAL;
; 0767 2
; 0768 1      END;

```

	6D	0021	CF	DE	00002	.ENTRY PROBE_ADDRESS, Save nothing	; 0697
						MOVAL 4\$, (FP)	; 0733
						; Read the address contained in PROBE_ADDR. If it causes a machine check	
						; the condition handler routine (HANDLER) will be called by the Supervisor	
						; Condition Handler Facility (CHF). That routine will unwind the stack	
						; one frame so that the return will be from this routine.	
			0C	AC	D5 00007	TSTL INTERLOCK	; 0753
				10	12 0000A	BNEQ 2\$	; 0755
	06		08	AC	E9 0000C	BLBC LENGTH, 1\$	; 0756
	50		04	BC	D0 00010	MOVL @PROBE_ADDR, PROBE_LONG	
				0D	11 00014	BRB 3\$	
	50		04	BC	B0 00016	1\$: MOVW @PROBE_ADDR, PROBE_WORD	; 0757
				07	11 0001A	BRB 3\$	; 0753
	50			01	B0 0001C	2\$: MOVW #1, TEMP	; 0761
	04			50	58 0001F	ADAWI TEMP, @PROBE_ADDR	; 0762
				01	D0 00023	3\$: MOVL #1, R0	; 0766
					04 00026	RET	; 0768
					0000 00027	4\$: .WORD Save nothing	; 0733
				7E	D4 00029	CLRL -(SP)	
				5E	DD 0002B	PUSHL SP	
	7E		04	AC	7D 0002D	MOVQ 4(AP), -(SP)	
	FED7			03	FB 00031	CALLS #3, HANDLER	
					04 00036	RET	

; Routine Size: 55 bytes, Routine Base: CODE + 02E1


```

: 0769 1 GLOBAL ROUTINE XFER_SETUP (UBE_NUM,MAP_NUM,DP_NUM,ROTATE_DP) :NOVALUE =
: 0770 1
: 0771 1 !**
: 0772 1 ! FUNCTIONAL DESCRIPTION:
: 0773 1 !
: 0774 1 !     This routine sets up the UBI and a UBE for a block transfer.
: 0775 1 !
: 0776 1 ! FORMAL PARAMETERS:
: 0777 1 !
: 0778 1 !     UBE_NUM - logical number of desired UBE
: 0779 1 !     MAP_NUM - number of lowest map to be used
: 0780 1 !     DP_NUM - desired UBI data path
: 0781 1 !     ROTATE_DP - if 1, rotate DP numbers among maps
: 0782 1 !                  if 0, use DP_NUM in all maps
: 0783 1 !
: 0784 1 ! IMPLICIT INPUTS:
: 0785 1 !
: 0786 1 !     UBE_BUF_SIZE - buffer size in pages
: 0787 1 !     UBE_BUF_ADDR[.UBE_NUM] - base address of buffer
: 0788 1 !
: 0789 1 ! IMPLICIT OUTPUTS: NONE
: 0790 1 !
: 0791 1 ! COMPLETION CODES: NONE
: 0792 1 !
: 0793 1 ! SIDE EFFECTS:
: 0794 1 !
: 0795 1 !     The UBI under test and the desired UBE are ready for transfer.
: 0796 1 !--
: 0797 1
: 0798 2 BEGIN
: 0799 2
: 0800 2 REGISTER
: 0801 2     MAX_MAP: WORD,           ! A11
: 0802 2     BUF_ADDR,
: 0803 2     M,                   ! Map counter
: 0804 2     N,                   ! UBE number
: 0805 2     UBUS_ADDR: WORD,      ! Used to build the page frame/byte number
: 0806 2     TEMP1,                ! A general purpose temporary
: 0807 2     TEMP2,                ! A general purpose temporary
: 0808 2     TEMP3;                ! A general purpose temporary
: 0809 2
: 0810 2 MACRO
: 0811 2     UBUS_PF = UBUS_ADDR<9,7>x, ! Defines the Unibus page frame field
: 0812 2     UBUS_BO = UBUS_ADDR<0,9>x, ! Defines the Unibus byte number field
: 0813 2     HIGH_BITS = UBUS_ADDR<0,2>x; ! Defines 2 MSB's of Unibus PF field
: 0814 2
: 0815 2 CODECOMMENT
: 0816 2 ''
: 0817 2 'This sets N equal to UBE_NUM, causing each UBE reference to be to',
: 0818 2 'UBE[N].',
: 0819 2 ''
: 0820 3     BEGIN
: 0821 3     N = .UBE_NUM;
: 0822 2     END;
: 0823 2

```

```

: 0824 2 CODECOMMENT
: 0825 2 ''
: 0826 2 'Setup of the UBE BA (bus address register) is accomplished in the',
: 0827 2 'following manner. First, the seven low bits of the desired map entry',
: 0828 2 'number are placed into bits 9 through 15 of a general purpose register.',
: 0829 2 'Then, bits 0 to 8 of the address of the I/O buffer to be used are',
: 0830 2 'copied into bits 0 to 8 of the register. Then, bits 0 through 15 of',
: 0831 2 'the register are copied into the UBE BA, and the upper two bits of the',
: 0832 2 'map entry number are copied into bits 0 and 1 of UBE CR2.',
: 0833 2 ''
: 0834 3 BEGIN
: 0835 3   BUF_ADDR = .UBE_BUF_ADDR[.UBE_NUM];
: 0836 3   UBUS_PF = .MAP_NUM<0,7>;           ! PF field gets 7 LSB's of map number
: 0837 3   UBUS_BO = .BUF_ADDR<0,9>;         ! BO field gets 10 buffer byte number
: 0838 3   UBE_BA = .UBUS_ADDR;             ! UBE BA gets 16 LSB's of merged PF/BO
: 0839 3   HIGH_BITS = .MAP_NUM<7,2>;
: 0840 3   UBE_CR2 = .HIGH_BITS;           ! UBE CR2 gets 2 MSB's of map number
: 0841 2 END;
: 0842 2
: 0843 2 CODECOMMENT
: 0844 2 ''
: 0845 2 'Each map entry is now written with consecutive page frame numbers (as',
: 0846 2 'each IO buffer is contiguous). We start with the map entry specified',
: 0847 2 'by the formal parameter MAP_NUM, and end with MAP_NUM + MAP_QUAN - 1.',
: 0848 2 'Depending on the contents of the parameter ROTATE_DP, we either use the',
: 0849 2 'given data path number (in parameter DP_NUM) in all maps, or rotate',
: 0850 2 'the data path numbers sequentially among the maps.',
: 0851 2 ''
: 0852 3 BEGIN
: 0853 3   TEMP3 = 0;
: 0854 3   MAX_MAP = .MAP_NUM + .UBE_BUF_SIZE - 1;           ! A11
: 0855 3   IF .UBE_MODE[.N] NEQ 0                             ! A11
: 0856 3   THEN MAX_MAP = .MAX_MAP + 1;                       ! A11
: 0857 3   INCR M FROM .MAP_NUM TO .MAX_MAP DO              ! M11
: 0858 4     BEGIN
: 0859 4       TEMP1 = .M;                                   ! A8
: 0860 4       ! TEMP1 = .MAP_NUM + .M;                     ! D8
: 0861 4       TEMP2 = .BUF_ADDR<9,15> + .M - .MAP_NUM;    ! A8
: 0862 4       ! TEMP2 = .BUF_ADDR<9,15> + .M;             ! D8
: 0863 4       IF .ROTATE_DP
: 0864 4       THEN
: 0865 5         BEGIN
: 0866 5           MAP_SETUP(.TEMP1,.TEMP2,.UBE_MODE[.N],.TEMP3);
: 0867 5           TEMP3 = .TEMP3 + 1;
: 0868 5           IF .TEMP3 EQL 4                             ! M8
: 0869 5           THEN
: 0870 5             TEMP3 = 0;
: 0871 5           END
: 0872 4       ELSE
: 0873 4         MAP_SETUP(.TEMP1,.TEMP2,.UBE_MODE[.N],.DP_NUM);
: 0874 3       END;
: 0875 2     END;
: 0876 2
: 0877 2 CODECOMMENT
: 0878 2 ''

```

```
; 0879 2 'The final step required to set up the UBE is to load the cycle count',
; 0880 2 'register with the twos complement of the desired cycle count.',
; 0881 2 ''
; 0882 3 BEGIN
; 0883 3 UBE_XFER_SIZE[N] = .UBE_BUF_SIZE*256;
; 0884 3 UBE_CC = -.UBE_BUF_SIZE*256;
; 0885 2 END;
; 0886 2
; 0887 1 END;
```

```
                                07FC 00000 .ENTRY XFER_SETUP, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 0769
                                5A      0000G CF 9E 00002 MOVAB UBE_MODE, R10 ;
                                59      0000G CF 9E 00007 MOVAB UBE_BUF_SIZE, R9 ;
; This sets N equal to UBE_NUM, causing each UBE reference to be
; to UBE[N].
;
                                53      04 AC D0 0000C MOVL UBE_NUM, N ; 0821
; Setup of the UBE BA (bus address register) is accomplished in
; the
; following manner. First, the seven low bits of the desired map
; entry
; number are placed into bits 9 through 15 of a general purpose
; register.
; Then, bits 0 to 8 of the address of the I/O buffer to be used
; are
; copied into bits 0 to 8 of the register. Then, bits 0 through
; 15 of
; the register are copied into the UBE BA, and the upper two bits
; of the
; map entry number are copied into bits 0 and 1 of UBE CR2.
;
                                50      04 AC D0 00010 MOVL UBE_NUM, R0 ; 0835
                                52      0000GCF40 D0 00014 MOVL UBE_BUF_ADDR[R0], BUF_ADDR ;
                                09      08 AC F0 0001A INSV MAP_NUM, #9, #7, UBUS_ADDR ; 0836
                                50      52 F0 00020 INSV BUF_ADDR, #0, #9, UBUS_ADDR ; 0837
                                51      0000GCF43 D0 00025 MOVL UBE_BASE_ADDR[N], R1 ;
                                04 A1 50 B0 0002B MOVW UBUS_ADDR, 4(R1) ; 0838
                                54      07 EF 0002F EXTZV #7, #2, MAP_NUM, R4 ; 0839
                                50      54 F0 00035 INSV R4, #0, #2, UBUS_ADDR ;
                                54      00 EF 0003A EXTZV #0, #2, UBUS_ADDR, R4 ; 0840
                                0E A1 54 B0 0003F MOVW R4, 14(R1) ;
```

```
; Each map entry is now written with consecutive page frame numbers (as
; each IO buffer is contiguous). We start with the map entry specified
; by the formal parameter MAP_NUM, and end with MAP_NUM + MAP_Q
```


;

```

UAN - 1.
; Depending on the contents of the parameter ROTATE_DP, we either use the
; given data path number (in parameter DP_NUM) in all maps, or
; rotate
; the data path numbers sequentially among the maps.

```

[illegible]

; Routine Size: 181 bytes, Routine Base: CODE + 0318

```

: 0888 1 GLOBAL ROUTINE MAP_SETUP (MAP_NUM,BUF_PFN,OFFSET_MODE,DP_NUM) :NOVALUE =
: 0889 1
: 0890 1 !++
: 0891 1 ! FUNCTIONAL DESCRIPTION:
: 0892 1 !
: 0893 1 !     This routine sets up a selected map entry.  Each map entry looks like
: 0894 1 !     this:
: 0895 1 !
: 0896 1 !           31           25           22 21           14           0           ! M7
: 0897 1 !     +-----+-----+-----+-----+-----+-----+-----+-----+
: 0898 1 !     ! V !           ! O !           ! DP !           ! PFN           !       ! M7
: 0899 1 !     +-----+-----+-----+-----+-----+-----+-----+-----+
: 0900 1 !
: 0901 1 !     V - valid bit
: 0902 1 !     O - offset mode
: 0903 1 !     DP - data path number
: 0904 1 !     PFN - page frame number
: 0905 1 !
: 0906 1 ! FORMAL PARAMETERS:
: 0907 1 !
: 0908 1 !     MAP_NUM - desired map entry (0 through 511)
: 0909 1 !     BUF_PFN - page frame number of I/O buffer
: 0910 1 !     OFFSET_MODE - 1 for address offset mode enabled,
: 0911 1 !                   0 for address offset mode disabled
: 0912 1 !     DP_NUM - desired data path (0 through 3)
: 0913 1 !
: 0914 1 ! IMPLICIT INPUTS: NONE
: 0915 1 !
: 0916 1 ! IMPLICIT OUTPUTS: NONE
: 0917 1 !
: 0918 1 ! COMPLETION CODES: NONE
: 0919 1 !
: 0920 1 ! SIDE EFFECTS:
: 0921 1 !
: 0922 1 !     The selected map is set up as desired.
: 0923 1 ! --
: 0924 1
: 0925 2 BEGIN
: 0926 2
: 0927 2 REGISTER
: 0928 2     TEMP;                ! A general purpose temporary
: 0929 2
: 0930 2 MACRO
: 0931 2     MAP_ENTRY =
: 0932 2     (.UBI_UNDER_TEST
: 0933 2     + %X'800' + 4*MAP_NUM)% ,
: 0934 2     PFN = TEMP<0,15>%,
: 0935 2     V = TEMP<31,1>%,
: 0936 2     O = TEMP<25,1>%,
: 0937 2     DP = TEMP<21,2>% ;
: 0938 2
: 0939 2 CODECOMMENT
: 0940 2
: 0941 2 'Build the map entry into a temporary register, and then copy it to the',
: 0942 2 'actual selected map.  The buffer physical page frame number is put into',

```

```
; 0943 2 'the temporary, the V bit is set, the desired offset mode value written,',
; 0944 2 'and finally the selected data path number is inserted.',
; 0945 2 '':
; 0946 3 BEGIN
; 0947 3 PFN = .BUF_PFN;
; 0948 3 V = 1;
; 0949 3 O = .OFFSET_MODE;
; 0950 3 DP = .DP_NUM;
; 0951 3 MAP_ENTRY = .TEMP;
; 0952 2 END;
; 0953 2
; 0954 1 END;
```

```
0000 00000 .ENTRY MAP_SETUP, Save nothing ; 0888
;
; Build the map entry into a temporary register, and then copy
; it to the
; actual selected map. The buffer physical page frame number i
; s put into
; the temporary, the V bit is set, the desired offset mode valu
; e written,
; and finally the selected data path number is inserted.
;
50 0F 00 08 AC F0 00002 INSV BUF_PFN, #0, #15, TEMP ; 0947
50 01 1F 01 01 F0 00008 INSV #1, #31, #1, TEMP ; 0948
50 01 19 0C AC F0 0000D INSV OFFSET_MODE, #25, #1, TEMP ; 0949
50 02 15 10 AC F0 00013 INSV DP_NUM, #21, #2, TEMP ; 0950
51 51 04 AC D0 00019 MOVL MAP_NUM, R1 ;
0800 51 0000GDF41 DE 0001D MOVAL @UBI_UNDER_TEST[R1], R1 ;
50 D0 00023 MOVL TEMP, 2048(R1) ; 0951
04 00028 RET ; 0954
```

; Routine Size: 41 bytes, Routine Base: CODE + 03CD


```

: 0955 1 GLOBAL ROUTINE BUFFER_SETUP (BUF_NUM,DATA_PATTERN,MODE) :NOVALUE =
: 0956 1
: 0957 1 !**
: 0958 1 ! FUNCTIONAL DESCRIPTION:
: 0959 1 !
: 0960 1 !     This routine sets up a buffer for a UBE transfer. It writes the
: 0961 1 !     desired data pattern throughout the buffer. If the MODE parameter
: 0962 1 !     is non zero, one extra word is written in the next page. This is
: 0963 1 !     for byte offset mode.
: 0964 1 !
: 0965 1 ! FORMAL PARAMETERS:
: 0966 1 !
: 0967 1 !     BUF_NUM - number of buffer
: 0968 1 !     DATA_PATTERN - desired data pattern
: 0969 1 !     MODE - Non zero indicates byte offset
: 0970 1 !
: 0971 1 ! IMPLICIT INPUTS:
: 0972 1 !
: 0973 1 !     UBE_BUF_SIZE - size in pages of buffer
: 0974 1 !
: 0975 1 ! IMPLICIT OUTPUTS:
: 0976 1 !
: 0977 1 !     The desired buffer is written with DATA_PATTERN.
: 0978 1 !
: 0979 1 ! COMPLETION CODES: NONE
: 0980 1 !
: 0981 1 ! SIDE EFFECTS:
: 0982 1 !
: 0983 1 !--
: 0984 1
: 0985 2 BEGIN
: 0986 2
: 0987 2 REGISTER
: 0988 2     BUF_SIZE,                ! Buffer size
: 0989 2     W;                    ! Word counter
: 0990 2
: 0991 2 BIND
: 0992 2     BUFFER = (.UBE_BUF_ADDR[.BUF_NUM]); ! Defines address of buffer
: 0993 2
: 0994 2 MAP
: 0995 2     BUFFER: VECTOR[32768,WORD];      ! Buffer is a vector with word
: 0996 2                                     ! elements, maximum extent of 32k
: 0997 2
: 0998 2 CODECOMMENT
: 0999 2 ''
: 1000 2 'Write desired data pattern into each word of the selected data buffer.',
: 1001 2 ''
: 1002 3 BEGIN
: 1003 3     BUF_SIZE = 256*.UBE_BUF_SIZE;
: 1004 3     IF .MODE NEQ 0                ! A10
: 1005 3     THEN BUF_SIZE = .BUF_SIZE + 1; ! Add a word for byte offset ! A10
: 1006 3     INCR W FROM 0 TO (.BUF_SIZE-1) DO
: 1007 4         BEGIN
: 1008 4             BUFFER[.W] = .DATA_PATTERN;
: 1009 3         END;

```

ZZ-ECCBA-1.4 01-04
UBI SUBROUTINES
01-04

J 9
6-Sep-1985 Fiche 1 Frame J9 Sequence 113
6-Sep-1985 17:18:07 VAX-11 Bliss-32 V4.1-746 Page 34
6-Sep-1985 17:14:45 [LEMIEUX.ECCBA.RELEASE]ECCBA1.B32;3 (13)

; 1010 2 END;
; 1011 2
; 1012 1 END;

```

                                0004 00000
                                04 AC D0 00002
50                                0000GCF40 D0 00006
52
                                ;
                                ; Write desired data pattern into each word of the selected dat
                                a buffer.
                                ;
50      0000G CF                0C 08 78 0000C                ASHL #8, UBE_BUF_SIZE, BUF_SIZE                ; 1003
                                AC D5 00012                TSTL MODE                ; 1004
                                02 13 00015                BEQL 1$                ;
                                50 D6 00017                INCL BUF_SIZE                ; 1005
                                51 01 CE 00019 1$:          MNEGL #1, W                ; 1008
                                05 11 0001C                BRB 3$                ;
                                6241 08 AC B0 0001E 2$:      MOVW DATA_PATTERN, (R2)[W]                ;
F7      51                                50 F2 00023 3$:      AOBLSS BUF_SIZE, W, 2$                ; 1006
                                04 00027                RET                ; 1012
```

; Routine Size: 40 bytes, Routine Base: CODE + 03F6

```

: 1013 1 GLOBAL ROUTINE MAP_BUFFERS (NUM_BUF,MODE) : = ! M10
: 1014 1
: 1015 1 !**
: 1016 1 ! FUNCTIONAL DESCRIPTION:
: 1017 1 !
: 1018 1 ! This routine allocates available memory for I/O buffer use. It
: 1019 1 ! partitions the available memory into the desired number of buffers for
: 1020 1 ! use by the UBE Block Transfer Test. If the MODE parameter is non
: 1021 1 ! zero, one additional page per buffer is reserved because the last
: 1022 1 ! transfer will overflow into this page.
: 1023 1
: 1024 1 ! FORMAL PARAMETERS:
: 1025 1 !
: 1026 1 ! NUM_BUF - number of desired buffers
: 1027 1 ! MODE - Non zero if BYTE OFFSET selected.
: 1028 1
: 1029 1 ! IMPLICIT INPUTS: NONE
: 1030 1
: 1031 1 ! IMPLICIT OUTPUTS:
: 1032 1 !
: 1033 1 ! UBE_BUF_ADDR[N] - vector pointing to the base address of the buffer
: 1034 1 ! UBE_BUF_SIZE - contains the size of a buffer in pages (all buffers are
: 1035 1 ! equal in size)
: 1036 1 ! UBE_MAP_NO[N] - Base map number of the buffer
: 1037 1 ! UBE_PAGE_CNT - Total number of pages received from supervisor.
: 1038 1
: 1039 1 ! COMPLETION CODES: Returns the number of MAPS (or pages) that have been used.
: 1040 1 ! If there is not enough memory for all the requested buffers,
: 1041 1 ! return is made with value zero.
: 1042 1
: 1043 1 ! SIDE EFFECTS: NONE
: 1044 1
: 1045 1 !--
: 1046 1
: 1047 2 BEGIN
: 1048 2
: 1049 2 REGISTER
: 1050 2 BUF_SIZE,
: 1051 2 I,
: 1052 2 M,
: 1053 2 N;
: 1054 2
: 1055 2 CODECOMMENT
: 1056 2 ''
: 1057 2 'Start out by attempting to get 256 pages per buffer (this is the largest',
: 1058 2 'data transfer possible) if only using 1 UBE or try to use all the MAP',
: 1059 2 'registers if more than one UBE is being used.',
: 1060 2 ''
: 1061 3 BEGIN
: 1062 3 CASE .NUM_BUF FROM 1 TO 3 OF
: 1063 3 SET
: 1064 3 [1]: N = 256;
: 1065 3 [3]: N = 495;
: 1066 3 [INRANGE,OUTRANGE]: N = 496;
: 1067 3 TES;

```



```

: 1068 2      END;
: 1069 2
: 1070 2      CODECOMMENT
: 1071 2      ''
: 1072 2      'Try to get the maximum buffer possible. If the buffer does not exist,',
: 1073 2      'the supervisor will return the largest possible buffer. This buffer',
: 1074 2      'size must be calculated from the address limits of the buffer. This',
: 1075 2      'size must then be made to split evenly among the required buffers.',
: 1076 2      'If the buffers are not at least 1 page each, return is made with',
: 1077 2      'value 0.',
: 1078 2      ''
: 1079 3      BEGIN
: 1080 3      IF $DS_GETBUF(PAGCNT=.N,RETADR=UBE_BUF_ADDR[0]) NEQ SS$_NORMAL
: 1081 4      THEN BEGIN
: 1082 4          IF .UBE_BUF_ADDR[0] EQL %X'FFFFFFFF'
: 1083 4          THEN RETURN 0;
: 1084 4          N = (.UBE_BUF_ADDR[1] - .UBE_BUF_ADDR[0] + 1) / 512;
: 1085 4          UBE_PAGE_CNT = .N;          ! Save total number of pages taken
: 1086 4          WHILE (.N MOD .NUM_BUF) NEQ 0 DO
: 1087 4              N = .N - 1;
: 1088 4          IF .N LSS .NUM_BUF THEN RETURN 0;
: 1089 4          END
: 1090 3      ELSE UBE_PAGE_CNT = .N;
: 1091 2      END;
: 1092 2
: 1093 2      CODECOMMENT
: 1094 2      ''
: 1095 2      'Now check if byte offset mode is being used. If it is, the size of',
: 1096 2      'the buffers must be reduced by one page to allow for the last transfer',
: 1097 2      'to fall into its on buffer.',
: 1098 2      ''
: 1099 3      BEGIN
: 1100 3      IF .MODE NEQ 0                                ! A10
: 1101 3      THEN
: 1102 4          BEGIN
: 1103 4              N = .N - .NUM_BUF;          ! Allow extra page per buffer    ! A10
: 1104 4              IF .N LSS .NUM_BUF        ! A1.1
: 1105 4              THEN RETURN 0;            ! A1.1
: 1106 4              M = 1;                    ! Set a mode flag used below    ! A10
: 1107 4          END
: 1108 3      ELSE
: 1109 3          M = 0;
: 1110 2      END;
: 1111 2
: 1112 2      CODECOMMENT
: 1113 2      ''
: 1114 2      'Now we have the size in pages of the buffer area. Here we determine',
: 1115 2      'the actual size in pages of each individual buffer by dividing the',
: 1116 2      'size of the buffer area (which is now given by N) by the required',
: 1117 2      'number of buffers.',
: 1118 2      ''
: 1119 3      BEGIN
: 1120 3      BUF_SIZE = .N/.NUM_BUF;
: 1121 2      END;
: 1122 2

```

```

; 1123 2 CODECOMMENT
; 1124 2 ''
; 1125 2 'Now calculate the starting address of each buffer and store into the',
; 1126 2 'vector UBE_BUF_ADDR[I], where I is the UBE number. Also, store the',
; 1127 2 'buffer size in the location UBE_BUF_SIZE.',
; 1128 2 '';
; 1129 3 BEGIN
; 1130 3 UBE_MAP_NO[0] = 0; ! A10
; 1131 3 INCR I FROM 1 TO (.NUM_BUF-1) DO
; 1132 4 BEGIN
; 1133 4 UBE_BUF_ADDR[I] = .UBE_BUF_ADDR[0] + ((.BUF_SIZE + .M) * .I * 512); ! M10
; 1134 4 UBE_MAP_NO[I] = ((.BUF_SIZE + .M) * .I); ! A10
; 1135 3 END;
; 1136 3 UBE_BUF_SIZE = .BUF_SIZE;
; 1137 2 END;
; 1138 3 RETURN (.N + (.M * .NUM_BUF)) ! First unused map number ! A10
; 1139 1 END;

```

```
.EXTRN DS$GETBUF
```

```

57      0000G  CF  9E 00002      .ENTRY  MAP_BUFFERS, Save R2,R3,R4,R5,R6,R7      ; 1013
      0000G  CF  9E 00002      MOVAB   UBE_BUF_ADDR, R7      ;

```

```
; Start out by attempting to get 256 pages per buffer (this is
; the largest
; data transfer possible) if only using 1 UBE or try to use all
; the MAP
; registers if more than one UBE is being used.
```

02 0014	01 0006	04 000D	AC 0000C	CF 0000C	00007 1\$:	CASEL .WORD	NUM_BUF, #1, #2 3\$-1\$,- 2\$-1\$,- 4\$-1\$	: 1062
	52	01F0	8F 0C	3C 11	00012 00017	2\$: MOVZWL BRB	#496, N 5\$	: 1066
	52	0100	8F 05	3C 11	00019 0001E	3\$: MOVZWL BRB	#256, N 5\$	: 1064
	52	01EF	8F	3C	00020	4\$: MOVZWL	#495, N	: 1065

```

; Try to get the maximum buffer possible. If the buffer does not exist,
; the supervisor will return the largest possible buffer. This
; buffer
; size must be calculated from the address limits of the buffer. This
; size must then be made to split evenly among the required buffers.
; If the buffers are not at least 1 page each, return is made with
; value 0.

```

```

0084 7E 7C 00025 5$: CLRQ -(SP) ; 1080
      8F BB 00027 PUSHR #^M<R2,R7> ;

```

```

00000000G 9F 04 FB 0002B CALLS #4, a#DS$GETBUF ;
01 50 D1 00032 CMPL R0, #1 ;
38 13 00035 BEQL 9$ ;
FFFFFFFFFF 8F 67 D1 00037 CMPL UBE_BUF_ADDR, #-1 ; 1082
2D 13 0003E BEQL 8$ ;
50 04 A7 67 C3 00040 SUBL3 UBE_BUF_ADDR, UBE_BUF_ADDR+4, R0 ; 1084
52 50 00000200 8F C7 00047 INCL R0 ;
0000G CF 52 D0 0004F MOVL #512, R0, N ; 1085
7E 00 52 01 7A 00054 6$: EMUL #1, N, #0, -(SP) ; 1086
50 8E 04 AC 50 D5 0005F EDIV NUM_BUF, (SP)+, R0, R0 ;
04 13 00061 BEQL 7$ ;
52 D7 00063 DECL N ; 1087
ED 11 00065 BRB 6$ ;
04 AC 52 D1 00067 7$: CMPL N, NUM_BUF ; 1088
07 18 0006B BGEQ 10$ ;
54 11 0006D 8$: BRB 15$ ;
0000G CF 52 D0 0006F 9$: MOVL N, UBE_PAGE_CNT ; 1090
;
; Now check if byte offset mode is being used. If it is, the si
; ze of
; the buffers must be reduced by one page to allow for the last
; transfer
; to fall into its on buffer.
;
08 AC D5 00074 10$: TSTL MODE ; 1100
OF 13 00077 BEQL 11$ ;
04 AC C2 00079 SUBL2 NUM_BUF, N ; 1103
52 D1 0007D CMPL N, NUM_BUF ; 1104
40 19 00081 BLSS 15$ ;
51 01 D0 00083 MOVL #1, M ; 1106
02 11 00086 BRB 12$ ; 1100
51 D4 00088 11$: CLRL M ; 1109
;
; Now we have the size in pages of the buffer area. Here we de
; termine
; the actual size in pages of each individual buffer by dividin
; g the
; size of the buffer area (which is now given by N) by the requ
; ired
; number of buffers.
;
50 52 04 AC C7 0008A 12$: DIVL3 NUM_BUF, N, BUF_SIZE ; 1120
;
; Now calculate the starting address of each buffer and store i
; nto the
; vector UBE_BUF_ADDR[I], where I is the UBE number. Also, sto
; re the
; buffer size in the location UBE_BUF_SIZE.
;
56 50 0000G CF B4 0008F CLRW UBE_MAP_NO ; 1130
51 C1 00093 ADDL3 M, BUF_SIZE, R6 ; 1133
53 D4 00097 CLRL I ;
13 11 00099 BRB 14$ ;

```


ZZ-ECCBA-1.4 01-04
UBI SUBROUTINES
01-04

B 10

6-Sep-1985

Fiche 1

Frame B10

Sequence 118

6-Sep-1985 17:18:07

VAX-11 Bliss-32 V4.1-746

Page 39

6-Sep-1985 17:14:45

[LEMIEUX.ECCBA.RELEASE]ECCBA1.B32;3

(14)

55	56	53	C5	0009B	13\$:	MULL3	I, R6, R5	:
54	55	09	78	0009F		ASHL	#9, R5, R4	:
6743	54	67	C1	000A3		ADDL3	UBE_BUF_ADDR, R4, UBE_BUF_ADDR[1]	:
	0000GCF43	55	B0	000A8		MOVW	R5, UBE_MAP_NO[1]	: 1134
E8		04	AC	F2 000AE	14\$:	AOBLSS	NUM_BUF, 1, 13\$	: 1131
	0000G	50	D0	000B3		MOVL	BUF_SIZE, UBE_BUF_SIZE	: 1136
		04	AC	C4 000B8		MULL2	NUM_BUF, R1	: 1138
		52	C0	000BC		ADDL2	N, R1	:
		51	D0	000BF		MOVL	R1, R0	:
			04	000C2		RET		:
		50	D4	000C3	15\$:	CLRL	R0	: 1139
			04	000C5		RET		:

; Routine Size: 198 bytes, Routine Base: CODE + 041E

```

; 1140 1 GLOBAL ROUTINE UNIBUS_SIZER :NOVALUE = ! Unibus sizer routine.
; 1141 1
; 1142 1 !++
; 1143 1 ! FUNCTIONAL DESCRIPTION:
; 1144 1 !
; 1145 1 !     This routine is necessary to map the Unibus for Unibus memory. Any
; 1146 1 !     memory existing on the Unibus will preclude certain maps from being
; 1147 1 !     tested. This routine sizes the Unibus for memory from Unibus address
; 1148 1 !     0 through 757777 (octal) in increments of 4096 word addresses. The
; 1149 1 !     output of the routine is a table of useable UBI map numbers.
; 1150 1 !
; 1151 1 ! FORMAL PARAMETERS: NONE
; 1152 1 !
; 1153 1 ! IMPLICIT INPUTS: NONE
; 1154 1 !
; 1155 1 ! IMPLICIT OUTPUTS:
; 1156 1 !
; 1157 1 !     VALID_MAPS - a bitvector representing useable map entries.
; 1158 1 !     It has the following format:
; 1159 1 !
; 1160 1 !           495 494 ... 2 1 0
; 1161 1 !     +-----+
; 1162 1 !     ! ! ! ! ! ! !
; 1163 1 !     +-----+
; 1164 1 !
; 1165 1 !     where each element is either a one if the corresponding
; 1166 1 !     map entry is useable, or a zero if it is not
; 1167 1 !
; 1168 1 !     NUM_MAPS - a word containing the total number of useable maps
; 1169 1 !
; 1170 1 !     FREE_MAP - a vector containing four numbers representing the first
; 1171 1 !     four available maps
; 1172 1 !
; 1173 1 !
; 1174 1 ! COMPLETION CODES: NONE
; 1175 1 !
; 1176 1 ! SIDE EFFECTS: NONE
; 1177 1 !
; 1178 1 ! --
; 1179 1 !
; 1180 2 BEGIN
; 1181 2
; 1182 2 REGISTER
; 1183 2     I,
; 1184 2     M,
; 1185 2     N,
; 1186 2     PROBE_ADDR;
; 1187 2
; 1188 2 CODECOMMENT
; 1189 2     'First the bitvector is initialized.',
; 1190 2     '
; 1191 2     '
; 1192 3     BEGIN
; 1193 3     INCR M FROM 0 TO 511 DO
; 1194 3     VALID_MAPS[.M] = 0;

```

```

: 1195 2      END;
: 1196 2
: 1197 2      CODECOMMENT
: 1198 2      ''
: 1199 2      'The Unibus is sized here for memory by reading a location every',
: 1200 2      '4k word addresses from 0 to 757777 (octal). The routine PROBE_ADDRESS',
: 1201 2      'is used to poke each location. It returns a value of 1 if the address',
: 1202 2      'exists, and 0 if the address does not exist.',
: 1203 2      ''
: 1204 3      BEGIN
: 1205 3      M = 0;
: 1206 3      INCR N FROM 0 TO 30 DO
: 1207 4      BEGIN
: 1208 4      PROBE_ADDR = (.N*8192) OR .UNIBUS_SPACE;
: 1209 4      IF NOT PROBE_ADDRESS(.PROBE_ADDR,0,0)
: 1210 4      THEN
: 1211 5      BEGIN
: 1212 5      INCR X FROM 0 TO 15 DO
: 1213 6      BEGIN
: 1214 6      M = .M + .X;
: 1215 6      I = .N*16 + .X;
: 1216 6      VALID_MAPS[.I] = 1;
: 1217 5      END;
: 1218 4      END;
: 1219 3      END;
: 1220 3      NUM_MAPS = .M;
: 1221 2      END;
: 1222 2
: 1223 2      CODECOMMENT
: 1224 2      ''
: 1225 2      'Now find the first four available maps and store their numbers in',
: 1226 2      'FREE_MAP.',
: 1227 2      ''
: 1228 3      BEGIN
: 1229 3      M = 0;
: 1230 3      INCR N FROM 0 TO 511 DO
: 1231 4      BEGIN
: 1232 4      IF .VALID_MAPS[.N]
: 1233 4      THEN
: 1234 5      BEGIN
: 1235 5      IF .M LEQ 3
: 1236 5      THEN
: 1237 6      BEGIN
: 1238 6      FREE_MAP[.M] = .N;
: 1239 6      M = .M + 1;
: 1240 6      END
: 1241 5      ELSE
: 1242 5      EXITLOOP;
: 1243 4      END;
: 1244 3      END;
: 1245 2      END;
: 1246 2
: 1247 1      END;

```


Label	Offset	Hex	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	
-------	--------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--

; Routine Size: 107 bytes, Routine Base: CODE + 04E4

```

: 1248 1 GLOBAL ROUTINE ENCODER :NOVALUE =
: 1249 1
: 1250 1 !**
: 1251 1 ! FUNCTIONAL DESCRIPTION:
: 1252 1 !
: 1253 1 !     This routine encodes a sequence of numbers from 0 to 8 into
: 1254 1 !     Reed-Muller (16,5) codewords.
: 1255 1 !
: 1256 1 ! FORMAL PARAMETERS: NONE
: 1257 1 !
: 1258 1 ! IMPLICIT INPUTS: NONE
: 1259 1 !
: 1260 1 ! IMPLICIT OUTPUTS:
: 1261 1 !
: 1262 1 !     CODEWORDS - a table of codewords representing the information words
: 1263 1 !     0 through 8
: 1264 1 !
: 1265 1 ! COMPLETION CODES: NONE
: 1266 1 !
: 1267 1 ! SIDE EFFECTS: NONE
: 1268 1 !
: 1269 1 !--
: 1270 1
: 1271 2 BEGIN
: 1272 2
: 1273 2 REGISTER
: 1274 2     I,
: 1275 2     N,
: 1276 2     TEMP1: WORD,
: 1277 2     TEMP2: WORD;
: 1278 2
: 1279 2 CODECOMMENT
: 1280 2 ''
: 1281 2 'Set up the table CODEVECTORS with the 16-bit vectors which form the',
: 1282 2 'basis for the Reed-Muller (16,5) coding space.',
: 1283 2 ''
: 1284 3     BEGIN
: 1285 3     CODEVECTORS[0] = %X'FFFF';
: 1286 3     CODEVECTORS[1] = %X'5555';
: 1287 3     CODEVECTORS[2] = %X'3333';
: 1288 3     CODEVECTORS[3] = %X'0F0F';
: 1289 2     END;
: 1290 2
: 1291 2 CODECOMMENT
: 1292 2 ''
: 1293 2 'Create a table of Reed-Muller (16,5) codewords to represent the numbers',
: 1294 2 '0 through 8. This is done by multiplying CODEVECTORS[I] times bit I of',
: 1295 2 'the number and forming the binary sum of each product.',
: 1296 2 ''
: 1297 3     BEGIN
: 1298 3     INCR N FROM 0 TO 8 DO
: 1299 4         BEGIN
: 1300 4             TEMP1 = 0;
: 1301 4             INCR I FROM 0 TO 3 DO
: 1302 5                 BEGIN

```

```

; 1303 5      TEMP2 = .N<.I,1>;
; 1304 5      TEMP2 = .CODEVECTORS[I]*.TEMP2;
; 1305 5      TEMP1 = .TEMP2 XOR .TEMP1;
; 1306 4      END;
; 1307 4      CODEWORDS[N] = .TEMP1;
; 1308 3      END;
; 1309 2      END;
; 1310 2      END;
; 1311 1      END;

```

001C 00000 .ENTRY ENCODER, Save R2,R3,R4 ; 1248

; Set up the table CODEVECTORS with the 16-bit vectors which form the basis for the Reed-Muller (16,5) coding space.

```

0000G CF 5555FFFF 8F D0 00002      MOVL #1431699455, CODEVECTORS ; 1285
0000G CF 0F0F3333 8F D0 0000B      MOVL #252654387, CODEVECTORS+4 ; 1287

```

; Create a table of Reed-Muller (16,5) codewords to represent the numbers 0 through 8. This is done by multiplying CODEVECTORS[I] times bit I of the number and forming the binary sum of each product.

			53	D4	00014	CLRL	N		; 1298
			50	B4	00016	CLRW	TEMP1		; 1300
			52	D4	00018	CLRL	I		; 1301
54	53	01	52	EF	0001A	EXTZV	I, #1, N, R4		; 1303
		51	54	B0	0001F	MOVW	R4, TEMP2		
		51	42	A4	00022	MULW2	CODEVECTORS[I], TEMP2		; 1304
		50	51	AC	00028	XORW2	TEMP2, TEMP1		; 1305
	EB	52	03	F3	0002B	AOBLEQ	#3, I, 2\$		; 1301
		52	50	B0	0002F	MOVW	TEMP1, CODEWORDS[N]		; 1307
	DD	53	08	F3	00035	AOBLEQ	#8, N, 1\$		; 1298
			04	00039	RET				; 1311

; Routine Size: 58 bytes, Routine Base: CODE + 054F

; 1312 1 GLOBAL ROUTINE DECODER :NOVALUE =

; 1313 1
 ; 1314 1 !++
 ; 1315 1 ! FUNCTIONAL DESCRIPTION:

; 1316 1 !
 ; 1317 1 ! This routine decodes a table of possibly corrupted codewords. It
 ; 1318 1 ! first decodes the four MSBs of all of the words, and then it decodes
 ; 1319 1 ! the LSBs of all of the words. This routine decodes codewords of the
 ; 1320 1 ! Reed-Muller (16,5) coding space only.

; 1321 1 !
 ; 1322 1 ! The basis of the Reed-Muller (32,6) coding space comprises the vectors

; 1323 1 !
 ; 1324 1 ! I 1111111111111111
 ; 1325 1 ! V1 0101010101010101
 ; 1326 1 ! V2 0011001100110011
 ; 1327 1 ! V3 0000111100001111

; 1328 1 !
 ; 1329 1 ! A codeword is created with the following expression:

; 1330 1 !
 ; 1331 1 ! codeword = G0*I XOR G1*V1 XOR G2*V2 XOR G3*V3,

; 1332 1 !
 ; 1333 1 ! where G0 is the LSB of the word to encode, and G3 is the MSB.

; 1334 1 !
 ; 1335 1 ! Each codeword has 8 disjoint redundant relations due to the nature of
 ; 1336 1 ! the encoding vectors. Note that in V1, the redundant relations are
 ; 1337 1 ! the bit pairs <31:30>, <29:28>, <27:26>, and so on. In V2, the
 ; 1338 1 ! redundant relations are the bit pairs <31:29>, <30:28>, <27:25>, and
 ; 1339 1 ! so on. In V3, the redundant relations are the bit pairs <31:27>,
 ; 1340 1 ! <30:26>, <29:25>, and so on. Thus, there are three sets of redundant
 ; 1341 1 ! relations in each codeword; each bit is represented by 8 disjoint
 ; 1342 1 ! relations.

; 1343 1 !
 ; 1344 1 ! To decode a particular bit, we calculate the value of each of the 8
 ; 1345 1 ! redundant relations. The position (weight) of the bit in the decoded
 ; 1346 1 ! word determines which bit pairs to use. The original word (that is,
 ; 1347 1 ! the 'message' word) is represented as

; 1348 1 !
 ; 1349 1 ! +---+---+---+
 ; 1350 1 ! !G3!G2!G1!G0!
 ; 1351 1 ! +---+---+---+

; 1352 1 !
 ; 1353 1 ! The bits Gn are referred to as 'information' bits. The LSB has no
 ; 1354 1 ! redundant relations inherent, since the value of G0 in the codeword
 ; 1355 1 ! is represented by the product G0*I. The next bit, G1, has the
 ; 1356 1 ! redundant relations represented by the set of 16 bit pairs, where the
 ; 1357 1 ! bit pairs are adjacent. Here it is convenient to introduce the notion
 ; 1358 1 ! of pair separation. The pair separation for the redundant relations
 ; 1359 1 ! representing G1 is 1. For G2, it is 2 and for G3 it is 4.

; 1360 1 !
 ; 1361 1 ! After decoding the value of each redundant relation for a particular
 ; 1362 1 ! bit, the sum of all of these values is taken. If there was no
 ; 1363 1 ! corruption of the codeword, the sum of all of the values of redundant
 ; 1364 1 ! relations will be zero or 8 (0 if the original information bit was 0,
 ; 1365 1 ! 8 if the original information bit was 1). If there was some
 ; 1366 1 ! corruption of the codeword, then the sum will be in between 0 and 8.

```

; 1367 1 ! Therefore a majority decision is made: if the sum is 4 or more, then
; 1368 1 ! the value of the message bit (that is, Gn) is a one; if the sum is
; 1369 1 ! less than 8, it is a zero.
; 1370 1 !
; 1371 1 ! After decoding the three MSBs, the LSB is decoded by eliminating each
; 1372 1 ! of the terms G1*V1, G2*V2, and G3*V3 from the codeword. This leaves
; 1373 1 ! simply G0*I. With no corruption in the codeword, this term would be
; 1374 1 ! either 16 bits of ones or sixteen bits of zeros. However, if there
; 1375 1 ! was any corruption, then the number of bits set in this term would lie
; 1376 1 ! between 0 and 16. Again, a majority decision specifies the value of
; 1377 1 ! of the term, and thus G0. Greater than 7 bits set will indicate that
; 1378 1 ! G0 is a one, and 7 or fewer bits will indicate that G0 is a zero.
; 1379 1 !
; 1380 1 ! FORMAL PARAMETERS: NONE
; 1381 1 !
; 1382 1 ! IMPLICIT INPUTS:
; 1383 1 !
; 1384 1 ! CODEVECTORS - a table of the requisite vectors which form the
; 1385 1 ! basis of the Reed-Muller (16,5) coding space
; 1386 1 !
; 1387 1 ! NOISY_WORDS - table of nine possibly corrupt (noisy) words
; 1388 1 !
; 1389 1 ! IMPLICIT OUTPUTS:
; 1390 1 !
; 1391 1 ! DECODED_WORDS - table of the nine decoded words
; 1392 1 !
; 1393 1 ! COMPLETION CODES: NONE
; 1394 1 !
; 1395 1 ! SIDE EFFECTS: NONE
; 1396 1 !
; 1397 1 ! --
; 1398 1 !
; 1399 2 BEGIN
; 1400 2
; 1401 2 REGISTER
; 1402 2 A: WORD,
; 1403 2 B: WORD,
; 1404 2 C,
; 1405 2 D,
; 1406 2 INFO_BIT,
; 1407 2 M,
; 1408 2 N,
; 1409 2 PAIR_SEP,
; 1410 2 POP,
; 1411 2 POS,
; 1412 2 TEMP;
; 1413 2
; 1414 2 MACRO
; 1415 2 DECODED = (DECODED_WORDS[.N])<.INFO_BIT,1>x;
; 1416 2
; 1417 2 CODECOMMENT
; 1418 2 ''
; 1419 2 'Decode the 3 LSBs of each possibly corrupt word first. This is done',
; 1420 2 'for each of the nine words.',
; 1421 2 '';
```

```

: 1422 3 BEGIN
: 1423 3 INCR N FROM 0 TO 8 DO
: 1424 4 BEGIN
: 1425 4 DECODED_WORDS[.N] = 0;
: 1426 4 PAIR_SEP = 1;
: 1427 4 INCR INFO_BIT FROM 1 TO 3 DO
: 1428 5 BEGIN
: 1429 5 POP = 0;
: 1430 5 IF .INFO_BIT NEQ 1
: 1431 5 THEN
: 1432 5 PAIR_SEP = .PAIR_SEP ^ 1;
: 1433 5 DECR POS FROM 15 TO 0 DO
: 1434 6 BEGIN
: 1435 6 TEMP = .NOISY_WORDS[.N];
: 1436 6 IF .TEMP<.POS,1>
: 1437 6 THEN
: 1438 6 A = 1
: 1439 6 ELSE
: 1440 6 A = 0;
: 1441 6 POS = .POS - .PAIR_SEP;
: 1442 6 IF .TEMP<.POS,1>
: 1443 6 THEN
: 1444 6 B = 1
: 1445 6 ELSE
: 1446 6 B = 0;
: 1447 6 IF .A XOR .B
: 1448 6 THEN
: 1449 6 POP = .POP + 1;
: 1450 6 IF (.POS MOD (1 ^ .INFO_BIT)) NEQ 0
: 1451 6 THEN
: 1452 6 POS = .POS + .PAIR_SEP;
: 1453 5 END;
: 1454 5 IF .POP GTR 4
: 1455 5 THEN
: 1456 5 DECODED = 1;
: 1457 4 END;
: 1458 3 END;
: 1459 2 END;
: 1460 2
: 1461 2 CODECOMMENT
: 1462 2 'Now that the three LSBs of the words are decoded, the MSBs will be',
: 1463 2 'decoded',
: 1464 2 '':
: 1465 2
: 1466 3 BEGIN
: 1467 3 INCR N FROM 0 TO 8 DO
: 1468 4 BEGIN
: 1469 4 B = .NOISY_WORDS[.N];
: 1470 4 INCR M FROM 1 TO 3 DO
: 1471 5 BEGIN
: 1472 5 INFO_BIT = .M;
: 1473 5 A = .DECODED;
: 1474 5 A = .A*.CODEVECTORS[.M];
: 1475 5 B = .B XOR .A;
: 1476 4 END;

```



```

; 1477 4 POP = 0;
; 1478 4 INCR M FROM 0 TO 15 DO
; 1479 5 BEGIN
; 1480 5 IF .B<.M,1>
; 1481 5 THEN
; 1482 5 POP = .POP + 1;
; 1483 4 END;
; 1484 4 IF .POP GTR 8
; 1485 4 THEN
; 1486 5 BEGIN
; 1487 5 INFO_BIT = 0;
; 1488 5 DECODED = 1;
; 1489 4 END;
; 1490 3 END;
; 1491 2 END;
; 1492 2
; 1493 1 END;

```

	OFFC	00000		.ENTRY	DECODER, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-	; 1312
5B	0000G	CF 9E	00002	MOVAB	R11 DECODED_WORDS, R11	; ;
					; Decode the 3 LSBs of each possibly corrupt word first. This	
					is done	
					; for each of the nine words.	
					;	
		55	D4 00007	CLRL	N	; 1423
	6B45	D4 00009	1\$:	CLRL	DECODED_WORDS[N]	; 1425
52		01 D0 0000C		MOVL	#1, PAIR_SEP	; 1426
58		01 D0 0000F		MOVL	#1, INFO_BIT	; 1435
		53 D4 00012	2\$:	CLRL	POP	; 1429
01		58 D1 00014		CMPL	INFO_BIT, #1	; 1430
		03 13 00017		BEQL	3\$	;
52		02 C4 00019		MULL2	#2, PAIR_SEP	; 1432
01		58 78 0001C	3\$:	ASHL	INFO_BIT, #1, R9	; 1450
57		0F D0 00020		MOVL	#15, POS	;
54	0000GCF	45 3C 00023	4\$:	MOVZWL	NOISY_WORDS[N], TEMP	; 1435
54		57 E1 00029		BBC	POS, TEMP, 5\$	; 1436
50		01 B0 0002D		MOVW	#1, A	; 1438
		02 11 00030		BRB	6\$	;
		50 B4 00032	5\$:	CLRW	A	; 1440
57		52 C2 00034	6\$:	SUBL2	PAIR_SEP, POS	; 1441
54		57 E1 00037		BBC	POS, TEMP, 7\$	; 1442
51		01 B0 0003B		MOVW	#1, B	; 1444
		02 11 0003E		BRB	8\$	;
		51 B4 00040	7\$:	CLRW	B	; 1446
56		50 3C 00042	8\$:	MOVZWL	A, R6	; 1447
5A		51 3C 00045		MOVZWL	B, R10	;
56		5A C0 00048		ADDL2	R10, R6	;
02		56 E9 0004B		BLBC	R6, 9\$	;
		53 D6 0004E		INCL	POP	; 1449

```

7E      00      57      01 7A 00050 9$: EMUL #1, POS, #0, -(SP) ; 1450
56      56      8E      59 7B 00055 EDIV R9, (SP)+, R6, R6 ;
      56      03 13 0005C TSTL R6 ;
      57      52 C0 0005E ADDL2 PAIR_SEP, POS ; 1452
BF      57 F4 00061 10$: SOBGEQ POS, 4$ ; 1433
04      53 D1 00064 CMPL POP, #4 ; 1454
      07 15 00067 BLEQ 11$ ;
      6B45 DF 00069 PUSHAL DECODED_WORDS[N] ; 1456
      58 E2 0006C BBSS INFO_BIT, a(SP)+, 11$ ;
      03 F3 00070 11$: AOBLEQ #3, INFO_BIT, 2$ ; 1427
      08 F3 00074 AOBLEQ #8, N, 1$ ; 1423
;
; Now that the three LSBs of the words are decoded, the MSBs wi
; ll be
; decoded
;
      54 D4 00078 CLRL N ; 1467
51      0000GCF44 B0 0007A 12$: MOVW NOISY_WORDS[N], B ; 1469
55      01 D0 00080 MOVL #1, M ; 1473
52      55 D0 00083 13$: MOVL M, INFO_BIT ; 1472
      6B44 DF 00086 PUSHAL DECODED_WORDS[N] ; 1473
      52 EF 00089 EXTZV INFO_BIT, #1, a(SP)+, R6 ;
      56 B0 0008E MOVW R6, A ;
50      0000GCF45 A4 00091 MULW2 CODEVECTORS[M], A ; 1474
51      50 AC 00097 XORW2 A, B ; 1475
55      03 F3 0009A AOBLEQ #3, M, 13$ ; 1470
      53 D4 0009E CLRL POP ; 1477
      55 D4 000A0 CLRL M ; 1478
      55 E1 000A2 14$: BBC M, B, 15$ ; 1480
      53 D6 000A6 INCL POP ; 1482
      0F F3 000A8 15$: AOBLEQ #15, M, 14$ ; 1478
      53 D1 000AC CMPL POP, #8 ; 1484
      09 15 000AF BLEQ 16$ ;
      52 D4 000B1 CLRL INFO_BIT ; 1487
      6B44 DF 000B3 PUSHAL DECODED_WORDS[N] ; 1488
      52 E2 000B6 BBSS INFO_BIT, a(SP)+, 16$ ;
      08 F3 000BA 16$: AOBLEQ #8, N, 12$ ; 1467
      04 000BE RET ; 1493

```

; Routine Size: 191 bytes, Routine Base: CODE + 0589

ZZ-ECCBA-1.4 01-04
UBI SUBROUTINES
01-04

M 10
6-Sep-1985 6-Sep-1985 6-Sep-1985
Fiche 1 17:18:07 17:14:45
Frame M10 VAX-11 Bliss-32 V4.1-746
Sequence 129 [LEMIEUX.ECCBA.RELEASE]ECCBA1.B32;3
Page 50 (18)

; 1494 1 \$DS_ENDMOD ;
; 1495 1 END !End of module
; 1496 0 ELUDOM

.PSECT \$TSTCNT,NOWRT,NOEXE, OVR,2

00000 L_TSTCNT:
.BLKB 4

.EXTRN SYS\$UNWIND

PSECT SUMMARY

Name	Bytes	Attributes
CODE	1608	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$TSTCNT	4	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, OVR,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
SYS\$COMMON:[SYSLIB]STARLET.L32;3	10093	2	0	598	00:00.8
SYS\$COMMON:[SYSLIB]DIAG.L32;265	784	11	1	85	00:00.3

COMMAND QUALIFIERS

; BLISS/LIST ECCBA1.B32

; Size: 1608 code + 4 data bytes
; Run Time: 00:20.9
; Elapsed Time: 00:25.3
; Lines/CPU Min: 4300
; Lexemes/CPU-Min: 21539
; Memory Used: 143 pages
; Compilation Complete

Table of contents

(2)	89	DECLARATIONS
(3)	130	UET_ISR - UET Interrupt Service Routine
(4)	187	UET_ISR - UET Interrupt Service Routine (SECOND UNIBUS)
(5)	244	UBE0_ISR - UBE0 Interrupt Service Routine
(6)	286	UBE1_ISR - UBE1 Interrupt Service Routine
(7)	328	UBE2_ISR - UBE2 Interrupt Service Routine
(8)	370	UBE3_ISR - UBE3 Interrupt Service Routine

```
0000 1 .TITLE UBI_ISR UBI Interrupt Service Routines
0000 2 .IDENT /01-04/
0000 3
0000 4 ;
0000 5 ; COPYRIGHT (C) 1980
0000 6 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000 7 ;
0000 8 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000 9 ; COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
0000 10 ; ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000 11 ; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000 12 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000 13 ; TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
0000 14 ; REMAIN IN DEC.
0000 15 ;
0000 16 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 17 ; AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 18 ; CORPORATION.
0000 19 ;
0000 20 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 21 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000 22 ;
0000 23 ;
0000 24 ;++
0000 25 ; FACILITY: UBI Diagnostic Program
0000 26 ;
0000 27 ; ABSTRACT:
0000 28 ;
0000 29 ; This module contains the interrupt service routines supporting
0000 30 ; the UBI Diagnostic Program. This includes an interrupt service
0000 31 ; routine for the UET and four UBEs.
0000 32 ;
0000 33 ; ENVIRONMENT: Diagnostic Supervisor
0000 34 ;
0000 35 ; AUTHOR: Rand Gray, CREATION DATE: 7-Dec-78
0000 36 ;
0000 37 ; MODIFIED BY: Don Monroe
0000 38 ;
0000 39 ; 0-7 May 1979
0000 40 ; Changed the format of the UBI MAP registers.
0000 41 ;
0000 42 ; 0-8 July 1979
0000 43 ; Modified to support the real UET module.
0000 44 ;
0000 45 ; Fixed a BUG in XFER_SETUP routine. Loop that setup the maps
0000 46 ; would not work if MAP_NUM was not zero.
0000 47 ;
0000 48 ; 0-9 July 1979
0000 49 ; Changed references to 'Byte Offset' to 'Byte Number'
0000 50 ;
0000 51 ; 0-10 October 1979
0000 52 ; Added a return value to the MAP_BUFFERS routine to support
0000 53 ; the changes to Module 7 Revision 10.
0000 54 ; Added byte-offset parameter to MAP_BUFFERS routine so routine
0000 55 ; returns with one additional page per buffer.
```

ZZ-ECCBA-1.4 01-04
UBI_ISR
01-04

UBI Interrupt Service Routines

C 11

6-SEP-1985

Fiche 1 Frame C11

Sequence 132

6-SEP-1985 17:18:33 VAX/VMS Macro V04-00
13-FEB-1985 16:05:09 ECCBA2.MAR;4

Page 2
(1)

0000	56 ;		
0000	57 ;	0-11	October 22,1979
0000	58 ;		Added a routine to print MAP errors. Called via PRINT LINK
0000	59 ;		in error hard calls.
0000	60 ;		
0000	61 ;	0-12	December 7,1979
0000	62 ;		Fixed bug in PROBE_ADDRESS routine in the ADAWI builtin.
0000	63 ;		
0000	64 ;	0-13	April 7,1980
0000	65 ;		Changed address of unibus space so second UBI will work.
0000	66 ;		
0000	67 ;	0-14	April 8,1980
0000	68 ;		Added a print general routine and converted PRINTX after
0000	69 ;		error calls to print links.
0000	70 ;	1-00	June 16,1980
0000	71 ;		Initial Release
0000	72 ;	1-01	July 31,1980
0000	73 ;		Fixed bug in MAP_BUFFERS routine that would not detect
0000	74 ;		insuficient memory for the requested number of buffers.
0000	75 ;	1-02	December 29,1980
0000	76 ;		Fixed bug in MAP_BUFFERS routine. Supervisor allocates memory
0000	77 ;		even if not enough for requested buffer size.
0000	78 ;	1-03	May 6, 1981
0000	79 ;		Fixed bug in map_buffers routine.
0000	80 ;		
0000	81 ;		Kevin Lemieux
0000	82 ;		
0000	83 ;	1-04	February,1985
0000	84 ;		Fixed bug in INIT code. If two DW's were specified; after the
0000	85 ;		first pass one of them one of them was no longer tested.
0000	86 ;		Fixed minor bugs in TEST 29.
0000	87 ;--		


```
0000      89      .SBTTL DECLARATIONS
0000      90 ;
0000      91 ; INCLUDE FILES:
0000      92 ;
0000      93      .LIBRARY      /SYS$LIBRARY:DIAG/
0000      94 ;
0000      95 ; MACROS:
0000      96 ;
0000      97 ;
0000      98 ;
0000      99 ; EQUATED SYMBOLS:
0000     100 ;
0000     101 ;
0000     102 ;
0000     103 ; Bit numbers of BR bits in UET Control Register
0000     104 ;
00000008 0000     105      BR4 = 8
00000009 0000     106      BR5 = 9
0000000A 0000     107      BR6 = 10
0000000B 0000     108      BR7 = 11
0000     109 ;
0000     110 ;
0000     111 ; Bit names of bits in UBI CSR's
0000     112 ;
00000001 0000     113      PURGE = 1
0000     114 ;
0000     115 ;
0000     116 ; OWN STORAGE:
0000     117 ;
00000000 0000     118 UET_CTRL_REG: .LONG
0004     119 ;
0004     120 ;
0004     121 ; External referances
0004     122 ;
0004     123 ;
0004     124      $DS_BGNMOD      CEP_REPAIR
0004      ;;;      Macro-32 Diagnostic macro library Version 7.0 "DIAG.MLB (186)"
0004     125 ;
00000000 0004     126      .PSECT UBI_ISR, LONG, EXE, NOWRT
0000     127 ;
0000     128 ;
```

```

0000 130 .SBTTL UET_ISR - UET Interrupt Service Routine
0000 131 .ALIGN LONG
0000 132 ;++
0000 133 ; FUNCTIONAL DESCRIPTION:
0000 134 ;
0000 135 ; This routine is invoked by an interrupt from the UET. It clears the
0000 136 ; flag INTERRUPT_EXP, and sets the flag INTERRUPT_OCC. It also clears
0000 137 ; the bit in the UET Control Register that is forcing the interrupt.
0000 138 ;
0000 139 ; CALLING SEQUENCE:
0000 140 ;
0000 141 ; The UET vector must point to this routine.
0000 142 ;
0000 143 ; INPUT PARAMETERS: NONE
0000 144 ;
0000 145 ; IMPLICIT INPUTS:
0000 146 ;
0000 147 ; INTERRUPT_EXP - indicates that an interrupt was expected
0000 148 ;
0000 149 ; OUTPUT PARAMETERS: NONE
0000 150 ;
0000 151 ; IMPLICIT OUTPUTS:
0000 152 ;
0000 153 ; INTERRUPT_OCC - indicates that an interrupt occurred
0000 154 ;
0000 155 ; COMPLETION CODES: NONE
0000 156 ;
0000 157 ; SIDE EFFECTS: NONE
0000 158 ;
0000 159 ;--
0000 160 UET_ISR::
0000 161 PUSH R0,R1 ; Save R0 and R1
0000 162 MOVL 8(SP),R0 ; Get interrupt PC
0000 163 BBSC #0,INTERRUPT_EXP,10$ ; If interrupt was not expected,
0000 164 $DS_ERRSYS_S,UNIT=LUN,-
0000 165 PRLINK=PRINT_GENERAL,-
0000 166 P1=FMT_UNX_INT,P2=1,-
0000 167 P3=R0 ; then report system error,
0000 168 10$: BBSS #0,INTERRUPT_OCC,20$ ; Set interrupt occurred flag
0000 169 20$:
0000 170 UET_ISRX:
0000 171 MOVL UNIBUS_SPACE,UET_CTRL_REG
0000 172 ADDL #X3F464,UET_CTRL_REG
0000 173 ;
0000 174 ; Clear the bit in the UET Control Register corresponding to highest BR level.
0000 175 ;

```

```

50 00000000'FF B0 0051 176 MOVW @UET_CTRL_REG,R0 ; Get the interrupt bits
50 C000 8F AA 0058 177 BICW #^XC000,R0 ; Clear interrupt done bit and INIT
0C 50 0B E4 005D 178 BBSC #BR7,R0,30$ ; Branch if BR7 Set and Clear It
08 50 0A E4 0061 179 BBSC #BR6,R0,30$ ; Also BR6
04 50 09 E4 0065 180 BBSC #BR5,R0,30$ ; Also BR5
00 50 08 E4 0069 181 BBSC #BR4,R0,30$ ; Also BR4
00000000'FF 50 B0 006D 182 30$: MOVW R0,@UET_CTRL_REG ; Clear the bit in the CTRL reg
00000000'EF 00000006'EF 3C 0074 183 MOVZWL CHAN_STAT+6,INT_VECTOR ; Save Channel Status inase Interrup Test
03 BA 007F 184 POPR #^M<R0,R1> ; Unsave R0 and R1
02 0081 185 REI ; Return from interrupt

```



```

0082 187      .SBTTL  UET_ISR - UET Interrupt Service Routine (SECOND UNIBUS)
0082 188      .ALIGN  LONG
0084 189      ;**
0084 190      ; FUNCTIONAL DESCRIPTION:
0084 191      ;
0084 192      ; This routine is invoked by an interrupt from the UET. It clears the
0084 193      ; flag INTERRUPT_EXP, and sets the flag INTERRUPT_OCC. It also clears
0084 194      ; the bit in the UET Control Register that is forcing the interrupt.
0084 195      ;
0084 196      ; CALLING SEQUENCE:
0084 197      ;
0084 198      ; The UET vector must point to this routine.
0084 199      ;
0084 200      ; INPUT PARAMETERS: NONE
0084 201      ;
0084 202      ; IMPLICIT INPUTS:
0084 203      ;
0084 204      ; INTERRUPT_EXP - indicates that an interrupt was expected
0084 205      ;
0084 206      ; OUTPUT PARAMETERS: NONE
0084 207      ;
0084 208      ; IMPLICIT OUTPUTS:
0084 209      ;
0084 210      ; INTERRUPT_OCC - indicates that an interrupt occurred
0084 211      ;
0084 212      ; COMPLETION CODES: NONE
0084 213      ;
0084 214      ; SIDE EFFECTS: NONE
0084 215      ;
0084 216      ;--
0084 217      UET_ISR_2::
0084 218      PUSHR    #^M<R0,R1>          ; Save R0 and R1
0084 219      MOVL     8(SP),R0             ; Get interrupt PC
0084 220      BBSC     #0,INTERRUPT_EXP,10$ ; If interrupt was not expected,
0084 221      $DS_ERRSYS_S,UNIT=LUN,-
0084 222      PRLINK=PRINT_GENERAL,-
0084 223      P1=FMT_UNX_INT,P2=1,-
0084 224      P3=R0                          ; then report system error,
0084 225      PUSHL    R0
0084 226      PUSHL    1
0084 227      PUSHL    FMT_UNX_INT
0084 228      PUSHAL   PRINT_GENERAL
0084 229      PUSHL    #0
0084 230      PUSHL    LUN
0084 231      PUSHL    #SER
0084 232      ;;; GLOBAL ERROR 2
0084 233      CALLS    $$$M, a#DSSERRSYS
0084 234      #0,INTERRUPT_OCC,20$          ; Set interrupt occurred flag
0084 235      MOVL     UNIBUS_SPACE,UET_CTRL_REG
0084 236      ADDL     #^X3F466,UET_CTRL_REG
0084 237      ;
0084 238      ; Clear the bit in the UET Control Register corresponding to highest BR level.
0084 239      ;
0084 240      MOVW     aUET_CTRL_REG,R0      ; Get the interrupt bits
  
```

```

50 4000 8F AA 00DC 233 BICW #^X4000,R0 ; Clear interrupt done bit
OC 50 OC E4 00E1 234 BBSC #BR7+1,R0,30$ ; Branch if BR7 Set and Clear It
08 50 0B E4 00E5 235 BBSC #BR6+1,R0,30$ ; Also BR6
04 50 0A E4 00E9 236 BBSC #BR5+1,R0,30$ ; Also BR5
00 50 09 E4 00ED 237 BBSC #BR4+1,R0,30$ ; Also BR4
00000000'FF 50 B0 00F1 238 30$: MOVW R0,@UET_CTRL_REG ; Clear the bit in the CTRL reg
00000000'EF 00000006'EF 50 3C 00F8 239 MOVZWL CHAN_STAT+6,INT_VECTOR ; Save Channel Status inase Interrup Test
03 02 0103 240 POPR #^M<R0,R1> ; Unsave R0 and R1
0106 242 REI ; Return from interrupt

```

```

0106 244 .SBTTL UBE0_ISR - UBE0 Interrupt Service Routine
0106 245 .ALIGN LONG
0108 246 ;++
0108 247 ; FUNCTIONAL DESCRIPTION:
0108 248 ;
0108 249 ; This routine is invoked by an interrupt from UBE0. It calls the
0108 250 ; routine UBE_INTERRUPT, passing to that routine a zero, identifying
0108 251 ; the caller as UBE0_ISR.
0108 252 ;
0108 253 ; CALLING SEQUENCE:
0108 254 ;
0108 255 ; UBE0 vector must point to this routine.
0108 256 ;
0108 257 ; INPUT PARAMETERS: NONE
0108 258 ;
0108 259 ; IMPLICIT INPUTS: NONE
0108 260 ;
0108 261 ; OUTPUT PARAMETERS: NONE
0108 262 ;
0108 263 ; IMPLICIT OUTPUTS: NONE
0108 264 ;
0108 265 ; COMPLETION CODES: NONE
0108 266 ;
0108 267 ; SIDE EFFECTS: NONE
0108 268 ;
0108 269 ;--
0108 270
0108 271
0108 272
0108 273 UBE0_ISR::
0108 274 PUSH R0,R1 ; Save R0 and R1
50 00000000'EF 98 010A 275 CVTBL UBE_DP_NO,R0 ; Get the Data Path number in use
0108 276 BLSS 1$ ; Branch if rotating
51 00000000'EF 01 0111 277 MOVL UBI_UNDER_TEST,R1 ; Get address of UBI under test
6140 01 011A 278 MOVL #PURGE,(R1)[R0] ; Purge the data path
00000000'EF 01 011E 279 1$: PUSHL #0 ; Identify caller as UBE0_ISR
0108 280 CALLS #1,UBE_INTERRUPT ; Call UBE interrupt handler
0108 281
0108 282 UBE0_ISR_X:
03 BA 0127 283 POP R0,R1 ; Unsave R0 and R1
02 0129 284 REI ; Return from interrupt

```



```

012A 286 .SBTTL UBE1_ISR - UBE1 Interrupt Service Routine
012A 287 .ALIGN LONG
012C 288 ;++
012C 289 ; FUNCTIONAL DESCRIPTION:
012C 290 ;
012C 291 ; This routine is invoked by an interrupt from UBE1. It calls the
012C 292 ; routine UBE_INTERRUPT, passing to that routine a zero, identifying
012C 293 ; the caller as UBE1_ISR.
012C 294 ;
012C 295 ; CALLING SEQUENCE:
012C 296 ;
012C 297 ; UBE1 vector must point to this routine.
012C 298 ;
012C 299 ; INPUT PARAMETERS: NONE
012C 300 ;
012C 301 ; IMPLICIT INPUTS: NONE
012C 302 ;
012C 303 ; OUTPUT PARAMETERS: NONE
012C 304 ;
012C 305 ; IMPLICIT OUTPUTS: NONE
012C 306 ;
012C 307 ; COMPLETION CODES: NONE
012C 308 ;
012C 309 ; SIDE EFFECTS: NONE
012C 310 ;
012C 311 ;--
012C 312
012C 313
012C 314
012C 315 UBE1_ISR::
50 00000001'EF 03 BB 012C 316 PUSH R0,R1 ; Save R0 and R1
012E 317 CVTBL UBE_DP_NO+1,R0 ; Get the Data Path number in use
0135 318 BLSS 1$ ; Branch if rotating
51 00000000'EF 0B 19 0137 319 MOVL UBI_UNDER_TEST,R1 ; Get address of UBI under test
6140 01 D0 013E 320 MOVL #PURGE,(R1)[R0] ; Purge the data path
01 DD 0142 321 1$: PUSHL #1 ; Identify caller as UBE1_ISR
00000000'EF 01 FB 0144 322 CALLS #1,UBE_INTERRUPT ; Call UBE interrupt handler
014B 323
014B 324 UBE1_ISR_X:
03 BA 014B 325 POP R0,R1 ; Unsave R0 and R1
02 014D 326 REI ; Return from interrupt

```

```

014E 328 .SBTTL UBE2_ISR - UBE2 Interrupt Service Routine
014E 329 .ALIGN LONG
0150 330 ;**
0150 331 ; FUNCTIONAL DESCRIPTION:
0150 332 ;
0150 333 ; This routine is invoked by an interrupt from UBE2. It calls the
0150 334 ; routine UBE_INTERRUPT, passing to that routine a zero, identifying
0150 335 ; the caller as UBE2_ISR.
0150 336 ;
0150 337 ; CALLING SEQUENCE:
0150 338 ;
0150 339 ; UBE2 vector must point to this routine.
0150 340 ;
0150 341 ; INPUT PARAMETERS: NONE
0150 342 ;
0150 343 ; IMPLICIT INPUTS: NONE
0150 344 ;
0150 345 ; OUTPUT PARAMETERS: NONE
0150 346 ;
0150 347 ; IMPLICIT OUTPUTS: NONE
0150 348 ;
0150 349 ; COMPLETION CODES: NONE
0150 350 ;
0150 351 ; SIDE EFFECTS: NONE
0150 352 ;
0150 353 ;--
0150 354
0150 355
0150 356
0150 357 UBE2_ISR::
0150 358 PUSHF ; Save R0 and R1
0150 359 CMTBL UBE_DP_NO+2,R0 ; Get the Data Path number in use
0150 360 BLSS 1$ ; Branch if rotating
0150 361 MOVL UBI_UNDER_TEST,R1 ; Get address of UBI under test
0150 362 MOVL #PURGE,(R1)[R0] ; Purge the data path
0150 363 1$: PUSHL #2 ; Identify caller as UBE2_ISR
0150 364 CALLS #1,UBE_INTERRUPT ; Call UBE interrupt handler
0150 365
0150 366 UBE2_ISR_X:
0150 367 POPR #^M<R0,R1> ; Unsave R0 and R1
0150 368 REI ; Return from interrupt
  
```

```

0172 370 .SBTTL UBE3_ISR - UBE3 Interrupt Service Routine
0172 371 .ALIGN LONG
0174 372 ;++
0174 373 ; FUNCTIONAL DESCRIPTION:
0174 374 ;
0174 375 ; This routine is invoked by an interrupt from UBE3. It calls the
0174 376 ; routine UBE_INTERRUPT, passing to that routine a zero, identifying
0174 377 ; the caller as UBE3_ISR.
0174 378 ;
0174 379 ; CALLING SEQUENCE:
0174 380 ;
0174 381 ; UBE3 vector must point to this routine.
0174 382 ;
0174 383 ; INPUT PARAMETERS: NONE
0174 384 ;
0174 385 ; IMPLICIT INPUTS: NONE
0174 386 ;
0174 387 ; OUTPUT PARAMETERS: NONE
0174 388 ;
0174 389 ; IMPLICIT OUTPUTS: NONE
0174 390 ;
0174 391 ; COMPLETION CODES: NONE
0174 392 ;
0174 393 ; SIDE EFFECTS: NONE
0174 394 ;
0174 395 ;--
0174 396
0174 397
0174 398
0174 399 UBE3_ISR::
0174 400 PUSHR #^M<R0,R1> ; Save R0 and R1
0174 401 CVTBL UBE_DP_NO+3,R0 ; Get the Data Path number in use
0174 402 BLSS 1$ ; Branch if rotating
0174 403 MOVL UBI_UNDER_TEST,R1 ; Get address of UBI under test
0174 404 MOVL #PURGE,(R1)[R0] ; Purge the data path
0174 405 1$: PUSHL #3 ; Identify caller as UBE3_ISR
0174 406 CALLS #1,UBE_INTERRUPT ; Call UBE interrupt handler
0174 407
0174 408 UBE3_ISR_X:
0174 409 POPR #^M<R0,R1> ; Unsave R0 and R1
0174 410 REI ; Return from interrupt
0174 411
0174 412 $DS_ENDMOD
0174 413 .END

```



```

$$M      = 00000007
$$N      = 00000001
$$S      = FFFFFFFF
$ENV     = 00000001  G
$ER      = 00000003
$MO      = 00000001  G
$ST      = 00000000
$TN      = 00000000
BIT...   = 00000002
BR4      = 00000008
BR5      = 00000009
BR6      = 0000000A
BR7      = 0000000B
CEP_FUNCTIONAL = 00000000
CEP_REPAIR   = 00000001
CHAN_STAT    *****  X  02
DS$ERRSYS    *****  X  02
ENV$M_DOMAIN = 00000002
ENV$M_LEVEL  = 00000001
ENV$M_SUPER  = 000003FC
ENV$S_DOMAIN = 00000001
ENV$S_LEVEL  = 00000001
ENV$S_SUPER  = 00000008
ENV$V_DOMAIN = 00000001
ENV$V_LEVEL  = 00000000
ENV$V_SUPER  = 00000002
ENV$CPU      = 00000000
ENV$FUNCTIONAL = 00000000
ENV$REPAIR   = 00000001
ENV$SUPER    = 00000001
ENV$SYSTEM   = 00000001
FMT_UNIX_INT *****  X  02
INTERRUPT_EXP *****  X  02
INTERRUPT_OCC *****  X  02
INT_VECTOR   *****  X  02
LUN          *****  X  02
PRINT_GENERAL *****  X  02
PURGE        = 00000001
SEP_FUNCTIONAL = 00000002
SEP_REPAIR    = 00000003
SIZ...       = 00000008
UBE0_ISR     00000108 RG  02
UBE0_ISR_X   00000127 R   02
UBE1_ISR     0000012C RG  02
UBE1_ISR_X   0000014B R   02
UBE2_ISR     00000150 RG  02
UBE2_ISR_X   0000016F R   02
UBE3_ISR     00000174 RG  02
UBE3_ISR_X   00000193 R   02
UBE_DP_NO    *****  X  02
UBE_INTERRUPT *****  X  02
UBI_UNDER_TEST *****  X  02
UET_CTRL_REG 00000000 R   01
UET_ISR      00000000 RG  02
UET_ISRX     0000003B R   02

```

```

UET_ISR_2      00000084 RG  02
UNIBUS_SPACE *****  X  02

```

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000004 (4.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
UBI_ISR	00000196 (406.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	106	00:00:00.20	00:00:00.67
Command processing	831	00:00:00.59	00:00:01.22
Pass 1	718	00:00:01.98	00:00:03.95
Symbol table sort	0	00:00:00.02	00:00:00.02
Pass 2	27	00:00:00.65	00:00:00.88
Symbol table output	2	00:00:00.03	00:00:00.04
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1686	00:00:03.50	00:00:06.80

The working set limit was 2012 pages.
21387 bytes (42 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 57 non-local and 10 local symbols.
413 source lines were read in Pass 1, producing 13 object records in Pass 2.
12 pages of virtual memory were used to define 11 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
SYS\$COMMON:[SYSLIB]DIAG.MLB;953	6
SYS\$COMMON:[SYSLIB]STARLET.MLB;1	4
TOTALS (all libraries)	10

159 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

MACRO/LIST ECCBA2.MAR

```

: 0001 0  MODULE UBI_TESTS_1_5 ( ! UBI diagnostic program tests 1 to 5
: 0002 0      IDENT = '01-04'
: 0003 0      ) =
: 0004 1  BEGIN
: 0005 1
: 0006 1  !
: 0007 1  ! COPYRIGHT (C) 1980
: 0008 1  ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
: 0009 1  !
: 0010 1  ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
: 0011 1  ! COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
: 0012 1  ! ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
: 0013 1  ! MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
: 0014 1  ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
: 0015 1  ! TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
: 0016 1  ! REMAIN IN DEC.
: 0017 1  !
: 0018 1  ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
: 0019 1  ! AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
: 0020 1  ! CORPORATION.
: 0021 1  !
: 0022 1  ! DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
: 0023 1  ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
: 0024 1  !
: 0025 1  !
: 0026 1  ! **
: 0027 1  ! FACILITY:      VAX-11 Diagnostic Library
: 0028 1  !
: 0029 1  ! ABSTRACT:      This module contains tests 1 to 5 of the COMET
: 0030 1  !                  Unibus Interface Diagnostic Program.
: 0031 1  !
: 0032 1  ! ENVIRONMENT:    VAX-11 Diagnostic Supervisor
: 0033 1  !
: 0034 1  ! AUTHOR:         Rand Gray, CREATION DATE: 11-Dec-78
: 0035 1  !
: 0036 1  ! MODIFIED BY:    Don Monroe : May 1979
: 0037 1  !
: 0038 1  !      0-7      May 1979
: 0039 1  !                  Changed the format of the UBI MAP registers.
: 0040 1  !
: 0041 1  !      0-8      July 1979
: 0042 1  !                  Modified to support the real UET module.
: 0043 1  !
: 0044 1  !                  Fixed a BUG in XFER_SETUP routine. Loop that setup the maps
: 0045 1  !                  would not work if MAP_NUM was not zero.
: 0046 1  !
: 0047 1  !      0-9      July 1979
: 0048 1  !                  Changed references to 'Byte Offset' to 'Byte Number'
: 0049 1  !
: 0050 1  !
: 0051 1  !      0-10     October 1979
: 0052 1  !                  Added a return value to the MAP_BUFFERS routine to support
: 0053 1  !                  the changes to Module 7 Revision 10.
: 0054 1  !                  Added byte-offset parameter to MAP_BUFFERS routine so routine
: 0055 1  !                  returns with one additional page per buffer.

```


ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

01-04

C 12

6-Sep-1985

Fiche 1

Frame C12

Sequence 145

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 2

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(1)

:	0056	1	:		
:	0057	1	:		
:	0058	1	:	0-11	October 22,1979
:	0059	1	:		Added a routine to print MAP errors. Called via PRINT LINK
:	0060	1	:		in error hard calls.
:	0061	1	:		
:	0062	1	:	0-12	December 7,1979
:	0063	1	:		Fixed bug in PROBE_ADDRESS routine in the ADAWI builtin.
:	0064	1	:		
:	0065	1	:	0-13	April 7,1980
:	0066	1	:		Changed address of unibus space so second UBI will work.
:	0067	1	:		
:	0068	1	:	0-14	April 8,1980
:	0069	1	:		Added a print general routine and converted PRINTX after
:	0070	1	:		error calls to print links.
:	0071	1	:	1-00	June 16,1980
:	0072	1	:		Initial Release
:	0073	1	:	1-01	July 31,1980
:	0074	1	:		Fixed bug in MAP_BUFFERS routine that would not detect
:	0075	1	:		insuficient memory for the requested number of buffers.
:	0076	1	:	1-02	December 29,1980
:	0077	1	:		Fixed bug in MAP_BUFFERS routine. Supervisor allocates memory
:	0078	1	:		even if not enough for requested buffer size.
:	0079	1	:	1-03	May 6, 1981
:	0080	1	:		Fixed bug in map_buffers routine.
:	0081	1	:		
:	0082	1	:		Kevin LeMieux
:	0083	1	:		
:	0084	1	:	1-04	February,1985
:	0085	1	:		Fixed bug in INIT code. If two DW's were specified; after the
:	0086	1	:		first pass one of them one of them was no longer tested.
:	0087	1	:		Fixed minor bugs in TEST 29.
:			:		

```

: 0088 1 !
: 0089 1 ! TABLE OF CONTENTS:
: 0090 1 !
: 0091 1 !   CSR Test
: 0092 1 !   Map Data Bus Test
: 0093 1 !   Map Chip Select Test
: 0094 1 !   Map Address Bus Test
: 0095 1 !   Map Entry Test
: 0096 1 !
: 0097 1 !
: 0098 1 !
: 0099 1 ! INCLUDE FILES:
: 0100 1 !
: 0101 1 !
: 0102 1 LIBRARY 'SYS$LIBRARY:STARLET';
: 0103 1 LIBRARY 'SYS$LIBRARY:DIAG';
: 0104 1 !
: 0105 1 !
: 0106 1 ! MACROS:
: 0107 1 !
: 0108 1 !
: 0109 1 !
: 0110 1 ! EQUATED SYMBOLS:
: 0111 1 !
: 0112 1 !
: 0113 1 LITERAL
: 0114 1     ALL_ONES = %X'FFFFFFFF',
: 0115 1     ALL_ZEROS = 0,
: 0116 1     BR4 = 3,
: 0117 1     BR5 = 5,
: 0118 1     BR6 = 9,
: 0119 1     BR7 = 17,
: 0120 1     CSR_MASK = %X'E0000001',
: 0121 1
: 0122 1     DATI = 1,
: 0123 1     DATIP = 3,
: 0124 1     DATO = 5,
: 0125 1     DATOB = 7,
: 0126 1
: 0127 1     DIAG_CHK_MODE = %X'C0000000',
: 0128 1     DISABLE_ECC = %X'20000000',
: 0129 1     ENABLE_ECC = %X'20000000',
: 0130 1     NORMAL = %X'E0000000',
: 0131 1     ERR = BIT31,
: 0132 1     MAP_MASK = %X'82607FFF',
: 0133 1     MEMORY_CONT = %X'F20000',
: 0134 1     NON_XIST_NEXUS = %X'F40000',
: 0135 1     NXM = BIT30,
: 0136 1     PURGE = BIT0,
: 0137 1     UCE = BIT29;
: 0138 1 !
: 0139 1 ! OWN STORAGE:
: 0140 1 !
: 0141 1 ! OWN
: 0142 1     BYTE_PATTERN: VECTOR[4,BYTE], ! Data patterns loaded at run time

```

! M7

```

: 0143 1 STATUS0: SIGNED WORD, ! Used by the UET status macro
: 0144 1 STATUS1, ! Used by the UBI status macro
: 0145 1 WORD_PATTERN: VECTOR[5,WORD]; ! Data patterns loaded at run time
: 0146 1
: 0147 1
: 0148 1 ! EXTERNAL REFERENCES:
: 0149 1 !
: 0150 1 EXTERNAL ROUTINE
: 0151 1 DECODER,
: 0152 1 ENCODER,
: 0153 1 PRINT_GENERAL:NOVALUE,
: 0154 1 PROBE_ADDRESS;
: 0155 1
: 0156 1 EXTERNAL
: 0157 1 BUFFER_SETUP,
: 0158 1 CODEWORDS: VECTOR[9,WORD],
: 0159 1 UBI_UNDER_TEST,
: 0160 1 UNIBUS_SPACE,
: 0161 1 DECODED_WORDS: VECTOR[9],
: 0162 1 EVENT_FLAG: BITVECTOR[4],
: 0163 1 FMT_BDP_DATI,
: 0164 1 FMT_BDP_DATO,
: 0165 1 FMT_BDP_DATO,
: 0166 1 FMT_BYTE_OFF,
: 0167 1 FMT_CMI_UB,
: 0168 1 FMT_CPU_RW,
: 0169 1 FMT_DDP_DATI,
: 0170 1 FMT_DDP_DATO,
: 0171 1 FMT_DDP_DATO,
: 0172 1 FMT_DP_SEL,
: 0173 1 FMT_MAP_ADDR,
: 0174 1 FMT_MAP_BIT_SA0, ! M11
: 0175 1 FMT_MAP_BIT_SA1, ! A11
: 0176 1 FMT_MAP_CS,
: 0177 1 FMT_MAP_DB_SA0, ! M8
: 0178 1 FMT_MAP_DB_SA1, ! A8
: 0179 1 FMT_MAP_FAIL,
: 0180 1 FMT_MCHK,
: 0181 1 FMT-UA1,
: 0182 1 FMT_UB_CMI,
: 0183 1 FMT_UET_STAT,
: 0184 1 FREE_MAP: VECTOR[4,WORD],
: 0185 1 HANDLER,
: 0186 1 INTERRUPT_EXP: BYTE,
: 0187 1 INTERRUPT_OCC: BYTE,
: 0188 1 LUN: BYTE,
: 0189 1 MAP_BUFFERS,
: 0190 1 MAP_SETUP,
: 0191 1 NOISY_WORDS: VECTOR[9,WORD],
: 0192 1 NUM_UB,
: 0193 1 PRINT_CSR_ERR,
: 0194 1 PRINT_DCMF_ERR,
: 0195 1 PRINT_MAP_ERR, ! A11
: 0196 1 UBIMOD_BDP,
: 0197 1 UBIMOD_BYTE_OFF,

```



```

: 0198 1 UBIMOD_CMI,
: 0199 1 UBIMOD_CPU_RW,
: 0200 1 UBIMOD_CSR,
: 0201 1 UBIMOD_DDP,
: 0202 1 UBIMOD_DP_SEL,
: 0203 1 UBIMOD_MAP,
: 0204 1 UBIMOD_MAP_CS,
: 0205 1 UBIMOD_MAP_ADDR,
: 0206 1 UBIMOD_MAP_CTRL,
: 0207 1 UBIMOD-UA1,
: 0208 1 UBIMOD-UB_CMI,
: 0209 1 SCRATCH: VECTOR[128],
: 0210 1 UBE_BASE_ADDR: VECTOR[4],
: 0211 1 UBE_BUF_SIZE,
: 0212 1 ! UBE_ERROR, ! D8
: 0213 1 UBE_STAT_ERR: BITVECTOR[4],
: 0214 1 UBE0_ISR,
: 0215 1 UET_ISR,
: 0216 1 UNIBUS_SIZER,
: 0217 1 VALID_MAPS: BITVECTOR[512],
: 0218 1 XFER_SETUP;
: 0219 1
: 0220 1 MACRO
: 0221 1 !++
: 0222 1 ! This macro defines the address of the control and status register
: 0223 1 ! for the given buffered data path number.
: 0224 1 !--
: 0225 1 UBI_CSR(BDP) = (.UBI_UNDER_TEST + BDP*4)x,
: 0226 1
: 0227 1 !++
: 0228 1 ! This macro defines the address of a UBI map for the given map
: 0229 1 ! number.
: 0230 1 !--
: 0231 1 UBI_MAP(NUM) = (.UBI_UNDER_TEST + %X'800' + NUM*4)x; ! M8
: 0232 1
: 0233 1 !++
: 0234 1 ! These macros temporarily define the addresses of the (simulated) UET
: 0235 1 ! registers. They will be replaced for the real UET.
: 0236 1 !--
: 0237 1 MACRO ! A8
: 0238 1 UET_ADDR_REG = (.UNIBUS_SPACE OR %X'3F460')<0,16>x, ! A8 M12
: 0239 1 UET_DATA_REG = (.UNIBUS_SPACE OR %X'3F462')<0,16>x, ! A8 M12
: 0240 1 UET_DATA_REGB = (.UNIBUS_SPACE OR %X'3F462')<0,8>x, ! A8 M12
: 0241 1 UET_CTRL_REG = (.UNIBUS_SPACE OR %X'3F464')<0,16>x, ! A8 M12
: 0242 1
: 0243 1 !++
: 0244 1 ! This macro is used for checking the status of the UET. ! A8
: 0245 1 !--
: M 0246 1 UET_STATUS(CALLOUT) = ! A8
: M 0247 1 STATUS0 = .UET_CTRL_REG; ! Read the UET control register ! A8
: M 0248 1 STATUS0 = .STATUS0 AND %X'E1'; ! A8
: M 0249 1 IF .STATUS0 NEQ 0 ! A8
: M 0250 1 THEN ! A8
: M 0251 1 BEGIN ! A8
: M 0252 1 $DS_ERRHARD(UNIT=.LUN,

```

```

; M 0253 1          MSGADR=CALLOUT,
; M 0254 1          PRLINK=PRINT_GENERAL,
; M 0255 1          P1 = FMT_UET_STAT,
; M 0256 1          P2 = 2,
; M 0257 1          P3 = 0,
; M 0258 1          P4 = .STATUS0);      ! A8
; 0259 1          END;%;                  ! A8
; 0260 1          ! A8
MACRO 0261 1
; 0262 1          !++
; 0263 1          ! This macro is used for checking the status of the UBI.
; 0264 1          !--
; M 0265 1          UBI_STATUS(BDP,CALLOUT) =
; M 0266 1          STATUS1 = .UBI_CSR(BDP) AND CSR_MASK;
; M 0267 1          IF .STATUS1 LSS 0
; M 0268 1          THEN
; M 0269 1          $DS_ERRHARD(UNIT=.LUN,
; M 0270 1          MSGADR=CALLOUT,
; M 0271 1          PRLINK=PRINT_CSR_ERR,
; 0272 1          P1=BDP,P2=0,P3=.STATUS1);%;
; 0273 1
; 0274 1          ! Required Diagnostic Supervisor macros.
; 0275 1          !
; L 0276 1          $DS_BGNMOD(ENV=0,TEST=1);
; %PRINT: 1          Bliss-32 Diagnostic Macro Library version 7.0 "DIAG.L32 (57)"
; 0277 1          $DS_DSADEF;
; 0278 1          $DS_SECDEF(ALL,UBE);
; 0279 1
; 0280 1          BUILTIN
; 0281 1          ROT;                      ! This is so we can do ROTLs

```

TEST_001: Control and Status Register Test

\$DS_BGNTEST (SECTION=(DEFAULT,ALL),TEXT='Control and Status Register Test');

!++
TEST DESCRIPTION:

In this test the three Control and Status Registers (CSRs) for the Buffered Data Paths (BDPs) are tested.

These registers are organized as follows:

31	30	29	28	1	0
+-----+-----+-----+-----+-----+					
!Error! NXM !		UCE !		not used ! Purge !	
+-----+-----+-----+-----+-----+					

Bit 31 Error OR of bits 29 and 30. Read-only.

Bit 30 Non existent memory Set when NXM status is received from CMI memory. Write one to clear.

Bit 29 Uncorrectable error This bit is set when uncorrectable error status is recieved from CMI memory. Write one to clear.

Bit 0 Purge Always reads zero.

This test checks that the registers are readable, and that writing a one to the status bits results in the bits being cleared.

!--
ASSUMPTIONS:

REGISTER
CSR,
N;

INCR N FROM 1 TO 3 DO
BEGIN

! Do for all three CSR's

CODECOMMENT

'Call the probe address routine with the address of CSR(N). If',
'reading the CSR results in a machine check, report the error.',
'':

BEGIN

DO

! Scope loop starts here

BEGIN

IF NOT PROBE_ADDRESS(UBI_CSR(.N),1,0)
THEN

BEGIN

\$DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CMI,

; P 0336 6


```
; P 0337 6
; L 0338 6
; xPRINT:
; 0339 5
; 0340 5
; 0341 5
; 0342 4
; 0343 4
; 0344 3
; 0345 3
; 0346 3
; 0347 3
; 0348 3
; 0349 3
; 0350 3
; 0351 3
; 0352 4
; 0353 4
; 0354 4
; 0355 5
; 0356 5
; 0357 5
; 0358 5
; 0359 5
; 0360 5
; 0361 6
; 0362 6
; P 0363 6
; L 0364 6
; xPRINT:
; 0365 5
; 0366 5
; 0367 5
; 0368 4
; 0369 4
; 0370 3
; 0371 3
; 0372 2
; 0373 2
; 0374 1
```

PRLINK = PRINT_GENERAL,
P1 = FMT_MCHK,P2 = 0);

Test 1, Subtest 0, Error 1
END;

END
WHILE \$DS_INLOOP; ! Scope loop ends here
END;

CODECOMMENT
,,
'Now write ones to bits 29 and 30 of the CSR. Read the CSR,',
'checking that bits 0, 29, 30, and 31 are all zero.',
,,

BEGIN
DO ! Scope loop starts here
BEGIN
UBI_CSR(.N) = NXM + UCE;
CSR = .UBI_CSR(.N);
IF (CSR_MASK AND .CSR) NEQ 0
THEN
BEGIN
CSR = .CSR_MASK AND .CSR;
\$DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,PRLINK=PRINT_CSR_ERR, ! M7
P1=.N,P2=0,P3=.CSR);
Test 1, Subtest 0, Error 2
END;

END
WHILE \$DS_INLOOP; ! Scope loop ends here
END;

END;
\$DS_ENDTEST;

.TITLE UBI_TESTS_1_5
.IDENT \01-04\

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00000 \$:
00000000 00000003 00010

.ADDRESS P.AAA, P.AAB, P.AAC, TEST_001
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

74 53 20 64 6E 61 20 6C 6F 72 74 6E 6F 43 20 0001 00000 P.AAA:
54 20 72 65 74 73 69 67 65 52 20 73 75 74 61 00002 P.AAB:
00011

.WORD 1
.ASCII \ Control and Status Register Test\

ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_001: Control and Status Register Test
TEST_001: Control and Status Register Test

J 12

6-Sep-1985

Fiche 1

Frame J12

Sequence 152

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 9

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(3)

74 73 65
00000000

00020
00023
00024

P.AAC: .BLKB 1
.LONG 0

.PSECT \$OWN\$,NOEXE,2

00000 BYTE_PATTERN:

.BLKB 4

00004 STATUS0:

.BLKB 2

00006 .BLKB 2

00008 STATUS1:

.BLKB 4

0000C WORD_PATTERN:

.BLKB 10

DSA\$AL_APTMAIL= 65024
DSA\$GL_FLAGS= 65024
DSA\$GL_APTCOM= 65028
DSA\$GL_PASSES= 65032
DSA\$GL_UNITS= 65036
DSA\$GL_SECTNO= 65040
DSA\$GL_SID= 65044
DSA\$GL_MSGTYP= 65088
DSA\$GL_ERRNO= 65092
DSA\$GL_EVENT= 65096
DSA\$GL_SUBTNO= 65100
DSA\$GL_TESTNO= 65104
DSA\$GL_PASSNO= 65108
DSA\$GL_DEVLEN= 65112
DSA\$GL_DEVNAM= 65116
DSA\$GL_MSGPTR= 65128

.EXTRN DECODER, ENCODER
.EXTRN PRINT_GENERAL, PROBE_ADDRESS
.EXTRN BUFFER_SETUP, CODEWORDS
.EXTRN UBI_UNDER_TEST, UNIBUS_SPACE
.EXTRN DECODED_WORDS, EVENT_FLAG
.EXTRN FMT_BDP_DATI, FMT_BDP_DATO
.EXTRN FMT_BDP_DATO, FMT_BYTE_OFF
.EXTRN FMT_CMI_UB, FMT_CPU_RW
.EXTRN FMT_DDP_DATI, FMT_DDP_DATO
.EXTRN FMT_DDP_DATO, FMT_DP_SEL
.EXTRN FMT_MAP_ADDR, FMT_MAP_BIT_SAO
.EXTRN FMT_MAP_BIT_SAI
.EXTRN FMT_MAP_CS, FMT_MAP_DB_SAO
.EXTRN FMT_MAP_DB_SAI, FMT_MAP_FAIL
.EXTRN FMT_MCHK, FMT_UAI
.EXTRN FMT_UB_CMI, FMT_UET_STAT
.EXTRN FREE_MAP, HANDLER
.EXTRN INTERRUPT_EXP, INTERRUPT_OCC
.EXTRN LUN, MAP_BUFFERS
.EXTRN MAP_SETUP, NOISY_WORDS
.EXTRN NUM_UBE, PRINT_CSR_ERR
.EXTRN PRINT_DCMP_ERR, PRINT_MAP_ERR
.EXTRN UBIMOD_BDP, UBIMOD_BYTE_OFF
.EXTRN UBIMOD_CMI, UBIMOD_CPU_RW
.EXTRN UBIMOD_CSR, UBIMOD_DDP

```
.EXTRN  UBIMOD_DP_SEL, UBIMOD_MAP
.EXTRN  UBIMOD_MAP_CS, UBIMOD_MAP_ADDR
.EXTRN  UBIMOD_MAP_CTRL
.EXTRN  UBIMOD_UA1, UBIMOD_UB_CMI
.EXTRN  SCRATCH, UBE_BASE_ADDR
.EXTRN  UBE_BUF_SIZE, UBE_STAT_ERR
.EXTRN  UBE0_ISR, UET_ISR
.EXTRN  UNIBUS_SIZER, VALID_MAPS
.EXTRN  XFER_SETUP, DS$ERRHARD
.EXTRN  DS$INLOOP, DS$BREAK
```

```
.PSECT  TEST_001,NOWRT,2
```

```
.ENTRY  TEST_001, Save R2,R3,R4,R5,R6
MOVAB   a#DS$INLOOP, R6
MOVAB   a#DS$ERRHARD, R5
MOVL    #1, N
```

```
; 0282
;
;
; 0319
```

```
; Call the probe address routine with the address of CSR(N). I
; reading the CSR results in a machine check, report the error
```

```
54          53          02  78 00013
          7E          01  7D 00017
          0000GDF44  9F 0001A
          CF          03  FB 0001F
          1C          50  E8 00024
          7E          7E  7C 00027
          7E          7E  7C 00029
          7E          7E  D4 0002B
          0000G  CF  9F 0002D
          0000G  CF  9F 00031
          0000G  CF  9F 00035
          7E          CF  9A 00039
          65          01  DD 0003E
          66          0A  FB 00040
          CE          00  FB 00043
          50          50  E8 00046
```

```
1$:  ASHL    #2, N, R4
2$:  MOVQ    #1, -(SP)
    PUSHAB  aUBI_UNDER_TEST[R4]
    CALLS   #3, PROBE_ADDRESS
    BLBS    R0, 3$
    CLRQ    -(SP)
    CLRQ    -(SP)
    CLRL    -(SP)
    PUSHAB  FMT_MCHK
    PUSHAB  PRINT_GENERAL
    PUSHAB  UBIMOD_CMI
    MOVZBL  LUN, -(SP)
    PUSHL   #1
    CALLS   #10, DS$ERRHARD
3$:  CALLS   #0, DS$INLOOP
    BLBS    R0, 2$
;
; Now write ones to bits 29 and 30 of the CSR. Read the CSR,
; checking that bits 0, 29, 30, and 31 are all zero.
```

```
54          53          02  78 00049
50          54          CF  C1 0004D
          60  60000000  8F  D0 00053
          52          60  D0 0005A
          E0000001  8F  52  D3 0005D
          26  13 00064
          50  E0000001  9F  D2 00066
          52          50  CA 0006D
          7E          7E  7C 00070
          7E          7E  D4 00072
          52          52  DD 00074
          7E          7E  D4 00076
```

```
4$:  ASHL    #2, N, R4
    ADDL3   UBI_UNDER_TEST, R4, R0
    MOVL    #1610612736, (R0)
    MOVL    (R0), CSR
    BITL    CSR, #-536870911
    BEQL    5$
    MCOML   a#^XE0000001, R0
    BICL2   R0, CSR
    CLRQ    -(SP)
    CLRL    -(SP)
    PUSHL   CSR
    CLRL    -(SP)
```

```
; 0357
;
;
; 0358
; 0359
;
; 0362
;
; 0364
;
```


; Routine Size: 163 bytes, Routine Base: TEST_001 + 0000

```

; 0375 2 $DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Map Data Bus Test');
; 0376 2
; 0377 2
; 0378 2 !++
; 0379 2 ! TEST DESCRIPTION:
; 0380 2 !
; 0381 2 ! This tests the integrity of a portion of the map data bus (that
; 0382 2 ! portion of the data path to and from the map which is used for CMI
; 0383 2 ! read/writes). The map is made up of 256 X 4 RAM ICs, organized into
; 0384 2 ! two rows of 5 ICs. In this test, a pattern of all ones is written to
; 0385 2 ! a map in the first row of RAMs, and then to a map in the second row.
; 0386 2 ! Any bit that is clear in both rows is determined to be a stuck at zero
; 0387 2 ! fault (although it might also be a compound fault with that RAM bit
; 0388 2 ! stuck at zero and the chip select line also stuck). Then, a pattern
; 0389 2 ! of all zeroes is written to a RAM in each of the two rows. Any bit
; 0390 2 ! that is set in both rows is determined to be a stuck at one fault. If
; 0391 2 ! each of the nineteen bits can toggle from 1 to 0, the map data bus
; 0392 2 ! visible to the program is considered to be good.
; 0393 2 !
; 0394 2 ! ASSUMPTIONS:
; 0395 2 !
; 0396 2 ! Test 1
; 0397 2 !--
; 0398 2 REGISTER
; 0399 2 FLAG,
; 0400 2 MAP_ENTRY,
; 0401 2 MAP_0: BITVECTOR[32],
; 0402 2 MAP_256: BITVECTOR[32],
; 0403 2 N;
; 0404 2
; 0405 2 MACRO
; 0406 2 PFN = MAP_ENTRY<0,15>%, ! Defines page frame field ! M7
; 0407 2 V = MAP_ENTRY<31,1>%, ! Defines valid bit field ! A7
; 0408 2 O = MAP_ENTRY<25,1>%, ! Defines offset bit field ! M7
; 0409 2 DP = MAP_ENTRY<21,2>%; ! Defines data path select field
; 0410 2
; 0411 2 CODECOMMENT
; 0412 2 ''
; 0413 2 'Write ones to all writeable bits in map 0.',
; 0414 2 ''
; 0415 2
; 0416 3 BEGIN
; 0417 3
; 0418 3 N = 0; ! Selects map entry 0
; 0419 3 PFN = ALL_ONES;
; 0420 3 V = ALL_ONES; ! A7
; 0421 3 O = ALL_ONES; ! M7
; 0422 3 DP = ALL_ONES;
; 0423 3 UBI_MAP(.N) = .MAP_ENTRY;
; 0424 3
; 0425 2 END;
; 0426 2
; 0427 2 CODECOMMENT
; 0428 2 ''
; 0429 2 'Write ones to all writeable bits in map 256.',

```

```

; 0430 2  '';
; 0431 2
; 0432 3 BEGIN
; 0433 3
; 0434 3 N = 256;
; 0435 3 UBI_MAP(.N) = .MAP_ENTRY;
; 0436 3
; 0437 2 END;
; 0438 2
; 0439 2 CODECOMMENT
; 0440 2  '';
; 0441 2  'Read map 0 and store in MAP_0. Read map 256 and store in MAP_256.',
; 0442 2  '';
; 0443 2
; 0444 3 BEGIN
; 0445 3
; 0446 3 N = 0;
; 0447 3 MAP_0 = .UBI_MAP(.N);
; 0448 3 N = 256;
; 0449 3 MAP_256 = .UBI_MAP(.N);
; 0450 3
; 0451 2 END;
; 0452 2
; 0453 2 CODECOMMENT
; 0454 2  '';
; 0455 2  'Now inspect each bit in the data path select field of both saved maps.',
; 0456 2  'If the bit in question is never one in either map 0 or map 256, report',
; 0457 2  'that bit stuck at zero.',
; 0458 2  '';
; 0459 2
; 0460 3 BEGIN
; 0461 3
; 0462 3 INCR N FROM 21 TO 22 DO
; 0463 4 BEGIN
; 0464 4
; 0465 4 IF .MAP_0[.N] EQL 0
; 0466 4 THEN
; 0467 4 FLAG = 1
; 0468 4 ELSE
; 0469 4 FLAG = 0;
; 0470 4 IF .MAP_256[.N] EQL 0
; 0471 4 THEN
; 0472 4 FLAG = 1
; 0473 4 ELSE
; 0474 4 FLAG = 0;
; 0475 4 IF .FLAG
; 0476 4 THEN
; 0477 5 BEGIN
; P 0478 5 $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP,PRLINK=PRINT_MAP_ERR, ! M7 M11
; P 0479 5 P1=0,P2=UBI_MAP(0),P3=FMT_MAP_DB_SA0,P4=%X'600000', ! A11
; L 0480 5 P5=(NOT(.MAP_0 XOR .MAP_256)) AND %X'600000'); ! A11
; %PRINT: Test 2, Subtest 0, Error 1
; 0481 5 ! $DS_PRINTX(FMT_MAP_DB_SA0,.N); ! D11
; 0482 4 END;
; 0483 4

```



```

: 0484 3      END;
: 0485 3
: 0486 2      END;
: 0487 2
: 0488 2      CODECOMMENT
: 0489 2      ''
: 0490 2      'Now inspect each bit in the valid and byte-offset field of both saved maps.',
: 0491 2      'If the bit in question is never one in either map 0 or map 256, report',
: 0492 2      'that bit stuck at zero.',
: 0493 2      '';
: 0494 2
: 0495 3      BEGIN
: 0496 3
: 0497 3      INCR N FROM 25 TO 31 BY 6 DO
: 0498 4          BEGIN
: 0499 4
: 0500 4              IF .MAP_0[.N] EQL 0
: 0501 4                  THEN
: 0502 4                      FLAG = 1
: 0503 4                  ELSE
: 0504 4                      FLAG = 0;
: 0505 4                  IF .MAP_256[.N] EQL 0
: 0506 4                      THEN
: 0507 4                          FLAG = 1
: 0508 4                      ELSE
: 0509 4                          FLAG = 0;
: 0510 4                  IF .FLAG
: 0511 4                      THEN
: 0512 5                      BEGIN
: 0513 5                          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP,PRLINK=PRINT_MAP_ERR, ! M7 M11
: 0514 5                          P1=0,P2=UBI_MAP(0),P3=FMT_MAP_DB_SAO,P4=XX'82000000', ! A11
: 0515 5                          P5=(NOT(.MAP_0 XOR .MAP_256)) AND XX'82000000');! A11
: 0516 5                          Test 2, Subtest 0, Error 2
: 0517 5                          ! $DS_PRINTX(FMT_MAP_DB_SAO,.N);
: 0518 5                          ! D11
: 0519 5                          END;
: 0520 5
: 0521 3                      END;
: 0522 3
: 0523 2          END;
: 0524 2      CODECOMMENT
: 0525 2      ''
: 0526 2      'Now inspect each bit in the page frame field of both saved maps.',
: 0527 2      'If the bit in question is never one in either map 0 or map 256, report',
: 0528 2      'that bit stuck at zero.',
: 0529 2      '';
: 0530 2
: 0531 3      BEGIN
: 0532 3
: 0533 3      INCR N FROM 0 TO 14 DO
: 0534 4          BEGIN
: 0535 4
: 0536 4              IF .MAP_0[.N] EQL 0
: 0537 4                  THEN
: 0538 4                      FLAG = 1

```

2)
ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_002: Map Data Bus Test

TEST_002: Map Data Bus Test

C 13

6-Sep-1985

Fiche 1

Frame C13

Sequence 158

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 15

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(4)

```
; 0538 4      ELSE
; 0539 4      FLAG = 0;
; 0540 4      IF .MAP_256[.N] EQL 0
; 0541 4      THEN
; 0542 4          FLAG = 1
; 0543 4      ELSE
; 0544 4          FLAG = 0;
; 0545 4      IF .FLAG
; 0546 4      THEN
; 0547 5          BEGIN
; P 0548 5              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP,PRLINK=PRINT_MAP_ERR, ! M7 M11
; P 0549 5              P1=0,P2=UBI_MAP(0),P3=FMT_MAP_DB_SAO,P4=XX'7FFF', ! A11
; L 0550 5              P5=(NOT(.MAP_0 XOR .MAP_256)) AND XX'7FFF'); ! A11
; xPRINT:
; 0551 5      Test 2, Subtest 0, Error 3
; 0552 4      ! $DS_PRINTX(FMT_MAP_DB_SAO,.N); ! D11
; 0553 4      END;
; 0554 3      END;
; 0555 3      END;
; 0556 2      CODECOMMENT
; 0557 2      'Now, write zeros to all writeable bits in map 0.',
; 0558 2      '':
; 0559 2      '':
; 0560 2      BEGIN
; 0561 2      N = 0; ! Selects map entry 0
; 0562 2      PFN = ALL_ZEROS;
; 0563 2      V = ALL_ZEROS; ! A7
; 0564 2      O = ALL_ZEROS; ! M7
; 0565 2      DP = ALL_ZEROS;
; 0566 2      UBI_MAP(.N) = .MAP_ENTRY;
; 0567 2      END;
; 0568 2      CODECOMMENT
; 0569 2      'Write zeross to all writeable bits in map 256.',
; 0570 2      '':
; 0571 2      '':
; 0572 2      BEGIN
; 0573 2      N = 256; ! Selects map entry 256
; 0574 2      UBI_MAP(.N) = .MAP_ENTRY;
; 0575 2      END;
; 0576 2      CODECOMMENT
; 0577 2      'Read map 0 and store in MAP_0. Read map 256 and store in MAP_256.',
; 0578 2      '':
; 0579 2      '':
; 0580 2      BEGIN
; 0581 2
; 0582 2
; 0583 2
; 0584 2
; 0585 2
; 0586 2
; 0587 2
; 0588 2
; 0589 2
; 0590 2
; 0591 3
```

3
)
ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_002: Map Data Bus Test

TEST_002: Map Data Bus Test

D 13

6-Sep-1985

Fiche 1

Frame D13

Sequence 159

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 16

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(4)

```
; 0592 3
; 0593 3      N = 0;
; 0594 3      MAP_0 = .UBI_MAP(.N);
; 0595 3      N = 256;
; 0596 3      MAP_256 = .UBI_MAP(.N);
; 0597 3
; 0598 2      END;
; 0599 2
; 0600 2      CODECOMMENT
; 0601 2      ``
; 0602 2      'Now inspect each bit in the data path select field of both MAP_0 and',
; 0603 2      'MAP_256. If the bit in question is never zero in either map 0 or map',
; 0604 2      '256, then that bit is stuck at one.',
; 0605 2      ``;
; 0606 2
; 0607 3      BEGIN
; 0608 3
; 0609 3      INCR N FROM 21 TO 22 DO
; 0610 4          BEGIN
; 0611 4
; 0612 4              IF .MAP_0[.N] EQL 1
; 0613 4                  THEN
; 0614 4                      FLAG = 1
; 0615 4                  ELSE
; 0616 4                      FLAG = 0;
; 0617 4                  IF .MAP_256[.N] EQL 1
; 0618 4                      THEN
; 0619 4                          FLAG = 1
; 0620 4                      ELSE
; 0621 4                          FLAG = 0;
; 0622 4                  IF .FLAG
; 0623 4                      THEN
; 0624 5                      BEGIN
; 0625 5                          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP,PRLINK=PRINT_MAP_ERR, ! M7 M11
; 0626 5                          P1=0,P2=UBI_MAP(0),P3=FMT_MAP_DB_SA1,P4=0, ! A11
; 0627 5                          P5=((.MAP_0 XOR .MAP_256)) AND xX'600000'); ! A11
; 0628 5                          Test 2, Subtest 0, Error 4
; 0629 5                          ! $DS_PRINTX(FMT_MAP_DB_SA1,.N); ! D11
; 0630 4                          END;
; 0631 3                      END;
; 0632 3
; 0633 2                  END;
; 0634 2
; 0635 2      CODECOMMENT
; 0636 2      ``
; 0637 2      'Now inspect each bit in the valid and byte-offset field of both MAP_0 and',
; 0638 2      'MAP_256. If the bit in question is never zero in either map 0 or map',
; 0639 2      '256, then that bit is stuck at one.',
; 0640 2      ``;
; 0641 2
; 0642 3      BEGIN
; 0643 3
; 0644 3      INCR N FROM 25 TO 31 BY 6 DO
; 0645 4          BEGIN
```



```

: 0646 4
: 0647 4      IF .MAP_0[.N] EQL 1
: 0648 4      THEN
: 0649 4          FLAG = 1
: 0650 4      ELSE
: 0651 4          FLAG = 0;
: 0652 4      IF .MAP_256[.N] EQL 1
: 0653 4      THEN
: 0654 4          FLAG = 1
: 0655 4      ELSE
: 0656 4          FLAG = 0;
: 0657 4      IF .FLAG
: 0658 4      THEN
: 0659 5          BEGIN
: P 0660 5              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP,PRLINK=PRINT_MAP_ERR, ! M7 M11
: P 0661 5                  P1=0,P2=UBI_MAP(0),P3=FMT_MAP_DB_SA1,P4=0, ! A11
: L 0662 5                  P5=((.MAP_0 XOR .MAP_256)) AND xX'82000000'); ! A11
: xPRINT:
: 0663 5      Test 2, Subtest 0, Error 5
: 0664 4      ! $DS_PRINTX(FMT_MAP_DB_SA1,.N); ! D11
: 0665 4      END;
: 0666 3      END;
: 0667 3      END;
: 0668 2      END;
: 0669 2      CODECOMMENT
: 0670 2      ''
: 0671 2      'Now inspect each bit in the page frame field of both MAP_0 and',
: 0672 2      'MAP_256. If the bit in question is never zero in either map 0 or map',
: 0673 2      '256, then that bit is stuck at one.',
: 0674 2      '';
: 0675 2      BEGIN
: 0676 2      INCR N FROM 0 TO 14 DO
: 0677 3      BEGIN
: 0678 3      IF .MAP_0[.N] EQL 1
: 0679 3      THEN
: 0680 4          FLAG = 1
: 0681 4      ELSE
: 0682 4          FLAG = 0;
: 0683 4      IF .MAP_256[.N] EQL 1
: 0684 4      THEN
: 0685 4          FLAG = 1
: 0686 4      ELSE
: 0687 4          FLAG = 0;
: 0688 4      IF .FLAG
: 0689 4      THEN
: 0690 5          BEGIN
: 0691 5              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP,PRLINK=PRINT_MAP_ERR, ! M7 M11
: 0692 5                  P1=0,P2=UBI_MAP(0),P3=FMT_MAP_DB_SA1,P4=0, ! A11
: 0693 5                  P5=((.MAP_0 XOR .MAP_256)) AND xX'7FFF'); ! A11
: 0694 5      Test 2, Subtest 0, Error 6
: P 0695 5      ! $DS_PRINTX(FMT_MAP_DB_SA1,.N); ! D11
: P 0696 5
: L 0697 5
: xPRINT:
: 0698 5

```

5)
ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_002: Map Data Bus Test
TEST_002: Map Data Bus Test

F 13

6-Sep-1985

Fiche 1

Frame F13

Sequence 161

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 18

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(4)

; 0699 4
; 0700 4
; 0701 3
; 0702 3
; 0703 2
; 0704 2
; 0705 1
END;
END;
END;
\$DS_ENDTEST;

00000000' 00000000' 00000000' 00000000' 00018 \$:
00000000 00000003 00028

.PSECT DISPATCH,NOWRT,NOEXE,2

.ADDRESS P.AAD, P.AAE, P.AAF, TEST_002
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

54 20 73 75 42 20 61 74 61 44 20 70 61 4D 11 0002 00028 P.AAD:
74 73 65 0002A P.AAE:
00000000 0003C P.AAF:

.WORD 2
.ASCII <17>\Map Data Bus Test\
.LONG 0

.PSECT TEST_002,NOWRT,2

OFFC 00000

.ENTRY TEST_002, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0375
R11
MOVAB UBIMOD MAP, R11
MOVAB PRINT MAP_ERR, R10
MOVAB a#DS\$ERRHARD, R9
MOVAB UBI_UNDER_TEST, R8

5B 0000G CF 9E 00002
5A 0000G CF 9E 00007
59 00000000G 9F 9E 0000C
58 0000G CF 9E 00013

; Write ones to all writeable bits in map 0.

53 82607FFF 56 D4 00018
50 00 B846 8F C8 0001A
0800 C0 53 D0 00021
D0 00026

CLRL N ; 0418
BISL2 #-2107604993, MAP_ENTRY ; 0422
MOVAL aUBI_UNDER_TEST[N], R0 ; 0423
MOVL MAP_ENTRY, 2048(R0)

; Write ones to all writeable bits in map 256.

56 0100 8F 3C 0002B
50 00 B846 DE 00030
0800 C0 53 D0 00035

MOVZWL #256, N ; 0434
MOVAL aUBI_UNDER_TEST[N], R0 ; 0435
MOVL MAP_ENTRY, 2048(R0)

; Read map 0 and store in MAP_0. Read map 256 and store in MAP_256.

50 00 B846 D4 0003A
54 0800 C0 DE 0003C
56 0100 8F D0 00041
50 00 B846 3C 00046
55 0800 C0 DE 0004B
D0 00050

CLRL N ; 0446
MOVAL aUBI_UNDER_TEST[N], R0 ; 0447
MOVL 2048(R0), MAP_0
MOVZWL #256, N ; 0448
MOVAL aUBI_UNDER_TEST[N], R0 ; 0449
MOVL 2048(R0), MAP_256

6
2)
ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_002: Map Data Bus Test

TEST_002: Map Data Bus Test

G 13

6-Sep-1985

Fiche 1 Frame G13

Sequence 162

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 19

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(4)

```
;
;
;
; Now inspect each bit in the data path select field of both sa
; ved maps.
; If the bit in question is never one in either map 0 or map 25
; 6, report
; that bit stuck at zero.
;
50          54          57          15 D0 00055      1$:      MOVL      #21, N                ; 0462
01          57 EF 00058      57:      EXTZV      N, #1, MAP_0, R0                ; 0465
05          12 0005D      BNEQ      2$
52          01 D0 0005F      MOVL      #1, FLAG                ; 0467
02          11 00062      BRB       3$
52          D4 00064      2$:      CLRL      FLAG                ; 0469
57          EF 00066      3$:      EXTZV      N, #1, MAP_256, R0                ; 0470
05          12 0006B      BNEQ      4$
52          01 D0 0006D      MOVL      #1, FLAG                ; 0472
02          11 00070      BRB       5$
52          D4 00072      4$:      CLRL      FLAG                ; 0474
30          52 E9 00074      5$:      BLBC      FLAG, 6$                ; 0475
7E          D4 00077      CLRL      -(SP)                ; 0480
50          54          55          55 CD 00079      XORL3      MAP_256, MAP_0, R0
7E 00600000 8F          50 CB 0007D      BICL3      R0, #6291456, -(SP)
00600000 8F          50 DD 00085      PUSHL      #6291456
0000G CF          9F 0008B      PUSHAB     FMT_MAP_DB_SAO
7E          68 00000800 8F          C1 0008F      ADDL3      #2048, UBI_UNDER_TEST, -(SP)
00000800 7E          D4 00097      CLRL      -(SP)
5A          DD 00099      PUSHL      R10
5B          DD 0009B      PUSHL      R11
7E          0000G CF          9A 0009D      MOVZBL     LUN, -(SP)
01          DD 000A2      PUSHL      #1
69          0A FB 000A4      CALLS      #10, DS$ERRHARD
AD          57          16 F3 000A7      6$:      AOBLEQ     #22, N, 1$                ; 0462
;
; Now inspect each bit in the valid and byte-offset field of bo
; th saved maps.
; If the bit in question is never one in either map 0 or map 25
; 6, report
; that bit stuck at zero.
;
50          54          57          19 D0 000AB      7$:      MOVL      #25, N                ; 0497
01          57 EF 000AE      57:      EXTZV      N, #1, MAP_0, R0                ; 0500
05          12 000B3      BNEQ      8$
52          01 D0 000B5      MOVL      #1, FLAG                ; 0502
02          11 000B8      BRB       9$
52          D4 000BA      8$:      CLRL      FLAG                ; 0504
57          EF 000BC      9$:      EXTZV      N, #1, MAP_256, R0                ; 0505
05          12 000C1      BNEQ      10$
52          01 D0 000C3      MOVL      #1, FLAG                ; 0507
02          11 000C6      BRB      11$
52          D4 000C8      10$:     CLRL      FLAG                ; 0509
30          52 E9 000CA      11$:     BLBC      FLAG, 12$                ; 0510
7E          D4 000CD      CLRL      -(SP)                ; 0515
50          54          55          55 CD 000CF      XORL3      MAP_256, MAP_0, R0
7E 82000000 8F          50 CB 000D3      BICL3      R0, #-2113929216, -(SP)
82000000 8F          50 DD 000DB      PUSHL      #-2113929216
;
```



```

                                0000G CF 9F 000E1 PUSHAB FMT_MAP_DB_SAO ;
                                00000800 8F C1 000E5 ADDL3 #2048, UBI_UNDER_TEST, -(SP) ;
7E 68 00000800 7E D4 000ED CLRL -(SP) ;
                                5A DD 000EF PUSHL R10 ;
                                5B DD 000F1 PUSHL R11 ;
                                7E 0000G CF 9A 000F3 MOVZBL LUN, -(SP) ;
                                02 DD 000F8 PUSHL #2 ;
                                69 0A FB 000FA CALLS #10, DS$ERRHARD ;
FFAB 57 06 1F F1 000FD 12$: ACBL #31, #6, N, 7$ ; 0497
;
; Now inspect each bit in the page frame field of both saved ma
ps.
; If the bit in question is never one in either map 0 or map 25
6, report
; that bit stuck at zero.
;
50 54 01 57 D4 00103 CLRL N ; 0532
                                57 EF 00105 13$: EXTZV N, #1, MAP_0, R0 ; 0535
                                05 12 0010A BNEQ 14$ ;
                                52 01 D0 0010C MOVL #1, FLAG ; 0537
                                02 11 0010F BRB 15$ ;
                                52 D4 00111 14$: CLRL FLAG ; 0539
                                57 EF 00113 15$: EXTZV N, #1, MAP_256, R0 ; 0540
                                05 12 00118 BNEQ 16$ ;
                                52 01 D0 0011A MOVL #1, FLAG ; 0542
                                02 11 0011D BRB 17$ ;
                                52 D4 0011F 16$: CLRL FLAG ; 0544
                                2F 52 E9 00121 17$: BLBC FLAG, 18$ ; 0545
                                7E D4 00124 CLRL -(SP) ; 0550
                                50 54 55 CD 00126 XORL3 MAP_256, MAP_0, R0 ;
                                7E 00007FFF 8F 50 CB 0012A BICL3 R0, #32767, -(SP) ;
                                7E 7FFF 8F 3C 00132 MOVZWL #32767, -(SP) ;
                                0000G CF 9F 00137 PUSHAB FMT_MAP_DB_SAO ;
                                00000800 8F C1 0013B ADDL3 #2048, UBI_UNDER_TEST, -(SP) ;
                                7E 5A DD 00145 CLRL -(SP) ;
                                5B DD 00147 PUSHL R10 ;
                                7E 0000G CF 9A 00149 PUSHL R11 ;
                                03 DD 0014E MOVZBL LUN, -(SP) ;
                                69 0A FB 00150 CALLS #10, DS$ERRHARD ;
AE 57 0E F3 00153 18$: AOBLEQ #14, N, 13$ ; 0532
;
; Now, write zeros to all writeable bits in map 0.
;
                                56 D4 00157 CLRL N ; 0565
                                53 82607FFF 8F CA 00159 BICL2 #-2107604993, MAP_ENTRY ; 0569
                                50 00 B846 DE 00160 MOVAL @UBI_UNDER_TEST[N], R0 ; 0570
0800 C0 53 D0 00165 MOVL MAP_ENTRY, 2048(R0) ;
;
; Write zeross to all writeable bits in map 256.
;
                                56 0100 8F 3C 0016A MOVZWL #256, N ; 0581
                                50 00 B846 DE 0016F MOVAL @UBI_UNDER_TEST[N], R0 ; 0582
0800 C0 53 D0 00174 MOVL MAP_ENTRY, 2048(R0) ;

```

; ;

50	55	01	02	11	001FC	BRB	27\$	:	
		01	52	D4	001FE	CLRL	FLAG	:	0651
			53	EF	00200	EXTZV	N, #1, MAP_256, R0	:	0652
			50	D1	00205	CMPL	R0, #1	:	
		52	05	12	00208	BNEQ	28\$	:	
			01	D0	0020A	MOVL	#1, FLAG	:	0654
			02	11	0020D	BRB	29\$	:	
		2C	52	D4	0020F	CLRL	FLAG	:	0656
			52	E9	00211	BLBC	FLAG, 30\$	:	0657
	50		7E	D4	00214	CLRL	-(SP)	:	0662
	7E	54	55	CD	00216	XORL3	MAP_256, MAP_0, R0	:	
		50	7DFFFFFF	8F	CB	BICL3	#2113929215, R0, -(SP)	:	
				7E	D4	00222	CLRL	-(SP)	
			0000G	CF	9F	00224	PUSHAB	FMT_MAP_DB_SA1	
	7E	68	00000800	8F	C1	00228	ADDL3	#2048, UBI_UNDER_TEST, -(SP)	
				7E	D4	00230	CLRL	-(SP)	
				5A	DD	00232	PUSHL	R10	
				5B	DD	00234	PUSHL	R11	
		7E	0000G	CF	9A	00236	MOVZBL	LUN, -(SP)	
				05	DD	0023B	PUSHL	#5	
		69		0A	FB	0023D	CALLS	#10, DS\$ERRHARD	
FFA9	53	06		1F	F1	00240	30\$: ACBL	#31, #6, N, 25\$	0644

; Now inspect each bit in the page frame field of both MAP_0 and
; MAP_256. If the bit in question is never zero in either map 0
; or map
; 256, then that bit is stuck at one.

50	54	01	53	D4	00246	CLRL	N	:	0679
		01	53	EF	00248	EXTZV	N, #1, MAP_0, R0	:	0682
			50	D1	0024D	CMPL	R0, #1	:	
		52	05	12	00250	BNEQ	32\$	:	
			01	D0	00252	MOVL	#1, FLAG	:	0684
			02	11	00255	BRB	33\$	:	
			52	D4	00257	CLRL	FLAG	:	0686
50	55	01	53	EF	00259	EXTZV	N, #1, MAP_256, R0	:	0687
		01	50	D1	0025E	CMPL	R0, #1	:	
			05	12	00261	BNEQ	34\$	:	
		52	01	D0	00263	MOVL	#1, FLAG	:	0689
			02	11	00266	BRB	35\$	:	
			52	D4	00268	CLRL	FLAG	:	0691
		29	52	E9	0026A	BLBC	FLAG, 36\$	:	0692
			7E	D4	0026D	CLRL	-(SP)	:	0697
	50	54	55	CD	0026F	XORL3	MAP_256, MAP_0, R0	:	
7E	50	0F	00	EF	00273	EXTZV	#0, #15, R0, -(SP)	:	
			7E	D4	00278	CLRL	-(SP)	:	
			0000G	CF	9F	0027A	PUSHAB	FMT_MAP_DB_SA1	
	7E	68	00000800	8F	C1	0027E	ADDL3	#2048, UBI_UNDER_TEST, -(SP)	
				7E	D4	00286	CLRL	-(SP)	
				5A	DD	00288	PUSHL	R10	
				5B	DD	0028A	PUSHL	R11	
		7E	0000G	CF	9A	0028C	MOVZBL	LUN, -(SP)	
				06	DD	00291	PUSHL	#6	
		69		0A	FB	00293	CALLS	#10, DS\$ERRHARD	

0
)
ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_002: Map Data Bus Test

TEST_002: Map Data Bus Test

K 13

6-Sep-1985

Fiche 1

Frame K13

Sequence 166

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 23

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(4)

AE 53
00000000G 9F
50

0E F3 00296 36\$:
00 FB 0029A
01 D0 002A1
04 002A4

AOBLEQ #14, N, 31\$
CALLS #0, a#DS\$BREAK
MOVL #1, R0
RET

; 0679
; 0703
; 0705
;

; Routine Size: 677 bytes, Routine Base: TEST_002 + 0000

; 0706 2
; 0707 2
; 0708 2
; 0709 2
; 0710 2
; 0711 2
; 0712 2
; 0713 2
; 0714 2
; 0715 2
; 0716 2
; 0717 2
; 0718 2
; 0719 2
; 0720 2
; 0721 2
; 0722 2
; 0723 2
; 0724 2
; 0725 2
; 0726 2
; 0727 2
; 0728 2
; 0729 2
; 0730 2
; 0731 2
; 0732 2
; 0733 2
; 0734 2
; 0735 2
; 0736 2
; 0737 2
; 0738 2
; 0739 2
; 0740 2
; 0741 2
; 0742 2
; 0743 2
; 0744 2
; 0745 2
; 0746 2
; 0747 2
; 0748 2
; 0749 2
; 0750 3
; 0751 3
; 0752 3
; 0753 3
; 0754 3
; 0755 3
; 0756 3
; 0757 3
; 0758 3
; 0759 2
; 0760 2

\$DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Map Chip Select Test');

!++
! TEST DESCRIPTION:

This test checks that the map chip select line is not stuck at one or zero. The map registers are made up of 256 X 4 RAMs, organized in two rows of 5 RAMs. The map chip select line is really a row select line, with the row select bit corresponding to the value of bit 10 of the map CMI address. To test whether this line is stuck, we write a unique value to a map entry with bit 10 at zero, and then we write a unique value to a map entry with bit 10 at one. If the chip select line is stuck at one or zero, the second value will overwrite the first. In order to guard against RAM data bit failures causing misleading results in this test, we write all ones to map entry 0, and all zeros to map entry 256, and then count the number of ones in each map upon reading. If the number of ones in a map is less than 9, the entry is considered to represent a zero. If the number of ones is 9 or more, the entry is considered to represent a one. In this way, the test tolerates up to 9 RAM data bit failures before the results of the test are misrepresented.

! ASSUMPTIONS:

! Test 1-Test 2

!--

REGISTER

MAP_ENTRY,
MAP_0: BITVECTOR[32],
N,
POPULATION;

MACRO

PFN = MAP_ENTRY<0,15>%,
V = MAP_ENTRY<31,1>%,
O = MAP_ENTRY<25,1>%,
DP = MAP_ENTRY<21,2>%;

! Defines page frame field ! M7
! Defines valid bit field ! A7
! Defines byte offset field ! M7
! Defines data path select field ! M7

CODECOMMENT

! Write all ones to map entry 0.
! :

BEGIN

N = 0;
PFN = ALL_ONES;
V = ALL_ONES;
O = ALL_ONES;
DP = ALL_ONES;
UBI_MAP(.N) = .MAP_ENTRY;

! Select map entry 0

! A7
! M7

END;

```

: 0761 2 CODECOMMENT
: 0762 2
: 0763 2 'Write all zeros to map entry 256.',
: 0764 2
: 0765 2
: 0766 3 BEGIN
: 0767 3
: 0768 3 N = 256; ! Select map entry 256
: 0769 3 PFN = ALL_ZEROS;
: 0770 3 V = ALL_ZEROS; ! A7
: 0771 3 O = ALL_ZEROS; ! M7
: 0772 3 DP = ALL_ZEROS;
: 0773 3 UBI_MAP(.N) = .MAP_ENTRY;
: 0774 3
: 0775 2 END;
: 0776 2 CODECOMMENT
: 0777 2
: 0778 2 'Now read map entry 0 and count the number of ones from bit 0 thru 14', ! M7
: 0779 2
: 0780 2
: 0781 2 BEGIN
: 0782 3
: 0783 3
: 0784 3 N = 0; ! Select map entry 0
: 0785 3 MAP_0 = .UBI_MAP(.N);
: 0786 3 POPULATION = 0; ! Initialize ones population counter
: 0787 3 INCR N FROM 0 TO 14 DO ! M7 M10
: 0788 4 BEGIN
: 0789 4
: 0790 4 IF .MAP_0[.N]
: 0791 4 THEN
: 0792 4 POPULATION = .POPULATION + 1;
: 0793 4
: 0794 3 END;
: 0795 3
: 0796 2 END;
: 0797 2 CODECOMMENT
: 0798 2
: 0799 2 'Now count the number of ones in bits 25 and 31.', ! M7
: 0800 2
: 0801 2
: 0802 2 BEGIN
: 0803 3
: 0804 3
: 0805 3 ! INCR N FROM 4 TO 5 DO ! D7
: 0806 3 ! BEGIN ! D7
: 0807 3
: 0808 3 ! IF .MAP_0[.N] ! D7
: 0809 3 ! THEN ! D7
: 0810 3 ! POPULATION = .POPULATION + 1; ! D7
: 0811 3
: 0812 3 ! END; ! D7
: 0813 3
: 0814 3
: 0815 3 IF .MAP_0[25] THEN POPULATION = .POPULATION + 1; ! A7

```


13
1)
ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_003: Map Chip Select Test

TEST_003: Map Chip Select Test

N 13

6-Sep-1985

Fiche 1

Frame N13

Sequence 169

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 26

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(5)

```
; 0816 3      IF .MAP_0[31] THEN POPULATION = .POPULATION + 1;      ! A7
; 0817 3
; 0818 2      END;
; 0819 2
; 0820 2 CODECOMMENT
; 0821 2      '
; 0822 2      'Finally, count the number of ones in bits 22 and 21.', ! M7
; 0823 2      '
; 0824 2
; 0825 3      BEGIN
; 0826 3
; 0827 3      INCR N FROM 21 TO 22 DO
; 0828 4          BEGIN
; 0829 4
; 0830 4          IF .MAP_0[.N]
; 0831 4          THEN
; 0832 4              POPULATION = .POPULATION + 1;
; 0833 4
; 0834 3          END;
; 0835 3
; 0836 2      END;
; 0837 2
; 0838 2 CODECOMMENT
; 0839 2      '
; 0840 2      'Now, decide how many ones were found overall. If there were 9 or more,',
; 0841 2      'then map entry 0 represents a one, and there is no error. If less than',
; 0842 2      '9 ones were found, map entry 0 represents a zero, and the chip select',
; 0843 2      'line is reported stuck at one or zero.',
; 0844 2      '
; 0845 2
; 0846 3      BEGIN
; 0847 3
; 0848 3      IF .POPULATION LSS 9
; 0849 3      THEN
; 0850 4          BEGIN
; P 0851 4              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP_CS,
; P 0852 4                  PRLINK = PRINT_GENERAL,
; L 0853 4                  P1 = FMT_MAP_CS,P2 = 0);
; %PRINT:      Test 3, Subtest 0, Error 1
; 0854 3          END;
; 0855 3
; 0856 2      END;
; 0857 2
; 0858 1      $DS_ENDTEST;
```

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00030 \$:
00000000 00000003 00040

.ADDRESS P.AAG, P.AAH, P.AAI, TEST_003
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

0003 00040 P.AAG: .WORD 3

22-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_003: Map Chip Select Test
TEST_003: Map Chip Select Test

B 14

6-Sep-1985

Fiche 1

Frame B14

Sequence 170

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 27

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(5)

```
63 65 6C 65 53 20 70 69 68 43 20 70 61 4D 14 00042 P.AAH: .ASCII <20>\Map Chip Select Test\ ;
74 73 65 54 20 74 00051 ;
00057 ;
00000000 00058 P.AAI: .BLKB 1 ;
.LONG 0 ;

.PSECT TEST_003,NOWRT,2
0004 00000 .ENTRY TEST_003, Save R2 ; 0706
; Write all ones to map entry 0.
;
51 D4 00002 CLRL N ; 0752
50 82607FFF 8F C8 00004 BISL2 #-2107604993, MAP_ENTRY ; 0756
52 0000GDF41 DE 0000B MOVAL @UBI_UNDER_TEST[N], R2 ; 0757
0800 C2 50 D0 00011 MOVL MAP_ENTRY, 2048(R2) ;
; Write all zeros to map entry 256.
;
51 0100 8F 3C 00016 MOVZWL #256, N ; 0768
50 82607FFF 8F CA 0001B BICL2 #-2107604993, MAP_ENTRY ; 0772
52 0000GDF41 DE 00022 MOVAL @UBI_UNDER_TEST[N], R2 ; 0773
0800 C2 50 D0 0002B MOVL MAP_ENTRY, 2048(R2) ;
; Now read map entry 0 and count the number of ones from bit 0
; thru 14
;
51 D4 0002D CLRL N ; 0784
50 0000GDF41 DE 0002F MOVAL @UBI_UNDER_TEST[N], R0 ; 0785
50 0800 C0 D0 00035 MOVL 2048(R0), MAP_0 ;
02 50 51 7C 0003A CLRQ POPULATION ; 0786
52 51 E1 0003C 1$: BBC N, MAP_0, 2$ ; 0790
F6 52 51 D6 00040 INCL POPULATION ; 0792
0E F3 00042 2$: AOBLEQ #14, N, 1$ ; 0787
; Now count the number of ones in bits 25 and 31.
;
02 50 19 E1 00046 BBC #25, MAP_0, 3$ ; 0815
51 D6 0004A INCL POPULATION ;
50 D5 0004C 3$: TSTL MAP_0 ; 0816
02 18 0004E BGEQ 4$ ;
51 D6 00050 INCL POPULATION ;
; Finally, count the number of ones in bits 22 and 21.
;
02 52 15 D0 00052 4$: MOVL #21, N ; 0827
50 52 E1 00055 5$: BBC N, MAP_0, 6$ ; 0830
51 D6 00059 INCL POPULATION ; 0832
F6 52 16 F3 0005B 6$: AOBLEQ #22, N, 5$ ; 0827
; Now, decide how many ones were found overall. If there were
; 9 or more,
; then map entry 0 represents a one, and there is no error. If
; less than
```

ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_003: Map Chip Select Test
TEST_003: Map Chip Select Test

C 14

6-Sep-1985 Fiche 1 Frame C14 Sequence 171
6-Sep-1985 17:18:43 VAX-11 Bliss-32 V4.1-746 Page 28
6-Sep-1985 17:14:50 [LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6 (5)

```
;
; 9 ones were found, map entry 0 represents a zero, and the chip select
; line is reported stuck at one or zero.
;
09          51 D1 0005F          CMPL      POPULATION, #9          ; 0848
          20 18 00062          BGEQ      7$          ;
          7E 7C 00064          CLRQ      -(SP)          ; 0853
          7E 7C 00066          CLRQ      -(SP)          ;
          7E D4 00068          CLRL      -(SP)          ;
          0000G CF 9F 0006A        PUSHAB   FMT_MAP_CS          ;
          0000G CF 9F 0006E        PUSHAB   PRINT_GENERAL        ;
          0000G CF 9F 00072        PUSHAB   UBIMOD_MAP_CS        ;
          7E 0000G CF 9A 00076      MOVZBL   LUN, -(SP)          ;
          01 DD 0007B          PUSHL      #1          ;
          00000000G 9F 0A FB 0007D      CALLS   #10, a#DS$ERRHARD    ;
          00000000G 9F 00 FB 00084 7$:  CALLS   #0, a#DS$BREAK      ; 0856
          50 01 D0 0008B          MOVL      #1, R0          ; 0858
          04 0008E          RET          ;
```

; Routine Size: 143 bytes, Routine Base: TEST_003 + 0000

; 0859 2
; 0860 2
; 0861 2
; 0862 2
; 0863 2
; 0864 2
; 0865 2
; 0866 2
; 0867 2
; 0868 2
; 0869 2
; 0870 2
; 0871 2
; 0872 2
; 0873 2
; 0874 2
; 0875 2
; 0876 2
; 0877 2
; 0878 2
; 0879 2
; 0880 2
; 0881 2
; 0882 2
; 0883 2
; 0884 2
; 0885 2
; 0886 2
; 0887 2
; 0888 2
; 0889 2
; 0890 2
; 0891 2
; 0892 2
; 0893 2
; 0894 2
; 0895 2
; 0896 2
; 0897 2
; 0898 2
; 0899 2
; 0900 2
; 0901 2
; 0902 2
; 0903 2
; 0904 2
; 0905 2
; 0906 2
; 0907 2
; 0908 2
; 0909 2
; 0910 2
; 0911 2
; 0912 2
; 0913 2

\$DS_BGNTTEST (SECTION=(DEFAULT,ALL),TEXT='Map Address Bus Test');

!++
! TEST DESCRIPTION:

This test checks that there are no map address bus bits stuck or shorted. A unique pattern is written into each map entry at the map entry addresses 0, 1, 2, 4, 8, 16, 32, 64, and 128 (so that each address bit assumes both a zero and a one). Reading each of these map entries after all patterns are written will reveal any stuck or shorted address bus bits, as the the last pattern written to a doubly addressed map will overwrite the earlier pattern written.

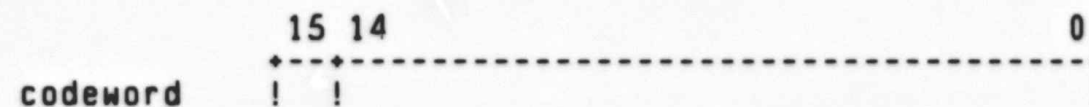
Consider the following example with map address bus bit 3 stuck at zero (the addresses are given in binary):

Map address bits 7:0	Data stored bits 23:9	Data retrieved bits 23:9
-----	-----	-----
00000000	0	4
00000001	1	1
00000010	2	2
00000100	3	3
00001000	4	4
00010000	5	5
00100000	6	6
01000000	7	7
10000000	8	8

Note that the contents of entry 0 were overwritten with the pattern stored at entry 8 (1000 base 2), since these two entries coincide when address bit 3 is stuck at zero. In general, an address bit n stuck at zero will show up as n+1 being stored in entry 0 in this scheme.

Unfortunately, a data bit fault in one of the RAMs in a particular map entry may point erroneously to an address bus bit fault. Therefore, to make this test insensitive to RAM data bit failures, it is necessary to encode each stored value prior to storing. The Reed-Muller (16,5) error-correcting code is used here.

Because the codewords we are using are sixteen bits long, and because there is no single field in a map entry having 16 contiguous bits, the codeword is stored in the map entry as follows:



```

0914 2  !
0915 2  !
0916 2  !
0917 2  !
0918 2  !
0919 2  !
0920 2  !
0921 2  ! ASSUMPTIONS:
0922 2  !
0923 2  ! Test 1-Test 3
0924 2  !
0925 2  !
0926 2  REGISTER
0927 2  MAP_ENTRY,
0928 2  M,
0929 2  N,
0930 2  TEMP;
0931 2
0932 2  MACRO
0933 2  PFN = MAP_ENTRY<0,15>X,          ! Defines page frame field      ! M7
0934 2  DPO = MAP_ENTRY<21,1>X,        ! Defines bit 21 of map entry   ! M7
0935 2  FIELD_1 = TEMP<0,15>X,         ! M7
0936 2  FIELD_2 = TEMP<15,1>X;         ! M7
0937 2
0938 2  CODECOMMENT
0939 2  'Generate the needed codewords by calling the routine ENCODER.',
0940 2  '':
0941 2  '':
0942 2  BEGIN
0943 3  ENCODER();
0944 3
0945 3  END;
0946 3
0947 2  CODECOMMENT
0948 2  'Write the encoded words representing the numbers 0 through 8 into map',
0949 2  'entries 0 through 8. Do this by first putting bits 15:1 of the encoded',
0950 2  'word into bits 15:0 of the register MAP_ENTRY, then putting bit 0 of',
0951 2  'the codeword into bit 21 of MAP_ENTRY. Then copy the contents of the',
0952 2  'register MAP_ENTRY into the actual map entry N. Do this for the map',
0953 2  'entries 0, 1, 2, 4, 8, 16, 32, 64, and 128.',
0954 2  '':
0955 2  '':
0956 2  BEGIN
0957 3  M = 0;
0958 3  N = 0;
0959 3  WHILE .N LEQ 128 DO
0960 4  BEGIN
0961 5  TEMP = .CODEWORDS[.M];
0962 5  PFN = .FIELD_1;
0963 5  DPO = .FIELD_2;

```

```

: 0969 4      UBI_MAP(.N) = .MAP_ENTRY;
: 0970 4      N = 1 ^ .M;      ! Let N = 2**M
: 0971 4      M = .M + 1;
: 0972 4
: 0973 3      END;
: 0974 3
: 0975 2      END;
: 0976 2
: 0977 2      CODECOMMENT
: 0978 2      ``
: 0979 2      'Now, retrieve all of the written map entries and store them in the',
: 0980 2      'array NOISY_WORDS.',
: 0981 2      ``;
: 0982 2
: 0983 3      BEGIN
: 0984 3
: 0985 3      M = 0;
: 0986 3      N = 0;
: 0987 3      WHILE .N LEQ 128 DO
: 0988 4          BEGIN
: 0989 4
: 0990 4          MAP_ENTRY = .UBI_MAP(.N);
: 0991 4          FIELD_1 = .PFN;
: 0992 4          FIELD_2 = .DPO;
: 0993 4          NOISY_WORDS[.M] = .TEMP;
: 0994 4          N = 1 ^ .M;      ! Let N = 2**M
: 0995 4          M = .M + 1;
: 0996 4
: 0997 3          END;
: 0998 3
: 0999 2      END;
: 1000 2
: 1001 2      CODECOMMENT
: 1002 2      ``
: 1003 2      'Call the decoder routine to decode the possibly corrupt (noisy)',
: 1004 2      'codewords. The decoder routine will store the decoded words in the',
: 1005 2      'table DECODED_WORDS. This should be a vector of longword elements',
: 1006 2      'representing the numbers 0 through 8. If there is an error in this',
: 1007 2      'sequence, then report an address bus bit failure.',
: 1008 2      ``;
: 1009 2
: 1010 3      BEGIN
: 1011 3      DECODER();
: 1012 3      INCR M FROM 0 TO 8 DO
: 1013 4          BEGIN
: 1014 4
: 1015 4          IF .DECODED_WORDS[.M] NEQ .M
: 1016 4          THEN
: 1017 5              BEGIN
: 1018 5                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP_ADDR,
: 1019 5                      PRLINK = PRINT_GENERAL,
: 1020 5                      P1 = FMT_MAP_ADDR,P2 = 1,P3 = .DECODED_WORDS[.M] - 1);
: 1021 4          END;
: 1022 4

```

Test 4, Subtest 0, Error 1

ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_004: Map Address Bus Test

TEST_004: Map Address Bus Test

G 14

6-Sep-1985

Fiche 1 Frame G14

Sequence 175

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 32

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(6)

; 1023 3
; 1024 3
; 1025 2
; 1026 2
; 1027 1
END;
END;
\$DS_ENDTEST;

00000000' 00000000' 00000000' 00000000' 00048 \$:
00000000 00000003 00058

.PSECT DISPATCH,NOWRT,NOEXE,2

.ADDRESS P.AAJ, P.AAK, P.AAL, TEST_004
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

75 42 20 73 73 65 72 64 64 41 20 70 61 4D 14 0004 0005C P.AAJ: .WORD 4
74 73 65 54 20 73 0005E P.AAK: .ASCII <20>\Map Address Bus Test\
0006D
00073
00000000 00074 P.AAL: .BLKB 1
.LONG 0

.PSECT TEST_004,NOWRT,2

001C 00000

.ENTRY TEST_004, Save R2,R3,R4

; 0859

; Generate the needed codewords by calling the routine ENCODER.

0000G CF

00 FB 00002

CALLS #0, ENCODER

; 0945

; Write the encoded words representing the numbers 0 through 8
into map
; entries 0 through 8. Do this by first putting bits 15:1 of t
he encoded
; word into bits 15:0 of the register MAP_ENTRY, then putting b
it 0 of
; the codeword into bit 21 of MAP_ENTRY. Then copy the content
s of the
; register MAP_ENTRY into the actual map entry N. Do this for
the map
; entries 0, 1, 2, 4, 8, 16, 32, 64, and 128.

00000080 8F

50
54
50

0F
53
01

52

0800

53
00
01
15
54
C4
01

0000GCF41
0000GDF42

51 7C 00007
52 D1 00009
28 14 00010
41 3C 00012
53 F0 00018
0F EF 0001D
54 F0 00022
42 DE 00027
50 D0 0002D
51 78 00032
51 D6 00036
CF 11 00038

1\$:

CLRQ M
CMPL N, #128
BGTR 2\$
MOVZWL CODEWORDS[M], TEMP
INSV TEMP, #0, #15, MAP_ENTRY
EXTZV #15, #1, TEMP, R4
INSV R4, #21, #1, MAP_ENTRY
MOVAL @UBI_UNDER_TEST[N], R4
MOVL MAP_ENTRY, 2048(R4)
ASHL M, #1, N
INCL M
BRB 1\$

; 0961
; 0963
;
; 0966
; 0967
; 0968
;
; 0969
;
; 0970
; 0971
; 0963

```

;
; Now, retrieve all of the written map entries and store them i
; n the
; array NOISY_WORDS.
;
;
; 53 0F 54 53 0F 52 00000080 8F 54 50 01 0F 0000GCF41 01 52 01
; 51 7C 0003A 2$: CLRQ M ; 0985
; 52 D1 0003C 3$: CMPL N, #128 ; 0987
; 28 14 00043 BGTR 4$ ;
; 42 DE 00045 MOVAL @UBI_UNDER_TEST[N], R4 ; 0990
; 0800 C4 D0 0004B MOVL 2048(R4), MAP_ENTRY ;
; 50 F0 00050 INSV MAP_ENTRY, #0, #15, TEMP ; 0991
; 15 EF 00055 EXTZV #21, #1, MAP_ENTRY, R4 ; 0992
; 54 F0 0005A INSV R4, #15, #1, TEMP ;
; 53 B0 0005F MOVW TEMP, NOISY_WORDS[M] ; 0993
; 51 78 00065 ASHL M, #1, N ; 0994
; 51 D6 00069 INCL M ; 0995
; CF 11 0006B BRB 3$ ; 0987
;
; Call the decoder routine to decode the possibly corrupt (nois
; y)
; codewords. The decoder routine will store the decoded words
; in the
; table DECODED_WORDS. This should be a vector of longword ele
; ments
; representing the numbers 0 through 8. If there is an error i
; n this
; sequence, then report an address bus bit failure.
;
;
; 0000G CF 00 FB 0006D 4$: CALLS #0, DECODER ; 1011
; 52 D4 00072 CLRL M ; 1012
; 52 0000GCF42 D1 00074 5$: CMPL DECODED_WORDS[M], M ; 1015
; 27 13 0007A BEQL 6$ ;
; 7E 7C 0007C CLRQ -(SP) ; 1020
; 7E D4 0007E CLRL -(SP) ;
; 01 C3 00080 SUBL3 #1, DECODED_WORDS[M], -(SP) ;
; 01 DD 00087 PUSHL #1 ;
; 0000G CF 9F 00089 PUSHAB FMT_MAP_ADDR ;
; 0000G CF 9F 0008D PUSHAB PRINT_GENERAL ;
; 0000G CF 9F 00091 PUSHAB UBIMOD_MAP_ADDR ;
; 7E 0000G CF 9A 00095 MOVZBL LUN, -(SP) ;
; 01 DD 0009A PUSHL #1 ;
; 0A FB 0009C 6$: CALLS #10, @DS$ERRHARD ;
; 08 F3 000A3 AOBLEQ #8, M, 5$ ; 1012
; 00 FB 000A7 CALLS #0, @DS$BREAK ; 1025
; 01 D0 000AE MOVL #1, R0 ; 1027
; 04 000B1 RET ;

```

; Routine Size: 178 bytes, Routine Base: TEST_004 + 0000

```

: 1028 2 $DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Map Entry Test');
: 1029 2
: 1030 2 !++
: 1031 2 ! TEST DESCRIPTION:
: 1032 2 !
: 1033 2 ! This tests that all readable/writeable bits of each map entry are
: 1034 2 ! not stuck at zero or one.
: 1035 2 !
: 1036 2 ! ASSUMPTIONS:
: 1037 2 !
: 1038 2 ! Test 1-Test 4
: 1039 2 !--
: 1040 2
: 1041 2 REGISTER
: 1042 2 MAP_ENTRY,
: 1043 2 N,
: 1044 2 TEMP;
: 1045 2
: 1046 2 MACRO
: 1047 2 PFN = MAP_ENTRY<0,15>x,
: 1048 2 V = MAP_ENTRY<31,1>x,
: 1049 2 O = MAP_ENTRY<25,1>x,
: 1050 2 DP = MAP_ENTRY<21,2>x;
: 1051 2
: 1052 2 CODECOMMENT
: 1053 2
: 1054 2 'Write zeros to every map entry.',
: 1055 2
: 1056 2
: 1057 3 BEGIN
: 1058 3
: 1059 3 PFN = ALL_ZEROS;
: 1060 3 V = ALL_ZEROS;
: 1061 3 O = ALL_ZEROS;
: 1062 3 DP = ALL_ZEROS;
: 1063 3 INCR N FROM 0 TO 511 DO
: 1064 4 BEGIN
: 1065 4
: 1066 4 UBI_MAP(.N) = .MAP_ENTRY;
: 1067 4
: 1068 3 END;
: 1069 3
: 1070 2 END;
: 1071 2
: 1072 2 CODECOMMENT
: 1073 2
: 1074 2 'Now read each map entry, checking that all bits are zero. Then write',
: 1075 2 'all ones to the map entry and read it, checking that all bits are one.',
: 1076 2
: 1077 2
: 1078 3 BEGIN
: 1079 3
: 1080 3 PFN = ALL_ONES;
: 1081 3 V = ALL_ONES;
: 1082 3 O = ALL_ONES;

```

! Defines page frame field ! M7
! Defines the valid bit field ! A7
! Defines byte offset bit field ! M7
! Defines data path select field! M7

! A7
! M7

! A7
! M7


```

: 1083 3      DP = ALL_ONES;
: 1084 3
: 1085 3      INCR N FROM 0 TO 511 DO
: 1086 4          BEGIN
: 1087 4
: 1088 4      CODECOMMENT
: 1089 4      'Read each map entry.',
: 1090 4      '':
: 1091 4          BEGIN
: 1092 5
: 1093 5      DO
: 1094 5          ! Scope loop begins here
: 1095 6          BEGIN
: 1096 6
: 1097 6      TEMP = .UBI_MAP(.N) AND MAP_MASK;
: 1098 6      IF .TEMP NEQ 0
: 1099 6      THEN
: 1100 7          BEGIN
: 1101 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP,PRLINK=PRINT_MAP_ERR,! M7 M11
: 1102 7      P1=.N,P2=UBI_MAP(.N),P3=FMT_MAP_BIT_SA0,! A11
: 1103 7      P4=0,P5=.TEMP); ! A11
: 1104 7      Test 5, Subtest 0, Error 1
: 1105 6      ! $DS_PRINTX(FMT_MAP_BIT_SA0,.N,UBI_MAP(.N),0,.TEMP); ! D11
: 1106 6      END;
: 1107 6      END
: 1108 5      WHILE $DS_INLOOP; ! Scope loop ends here
: 1109 5
: 1110 4      END;
: 1111 4
: 1112 4      CODECOMMENT
: 1113 4      'Now write all ones to every writeable bit in each map entry.',
: 1114 4      'then verify that all writeable bits are set.',
: 1115 4      '':
: 1116 4          BEGIN
: 1117 5
: 1118 5      DO
: 1119 5          ! Scope loop begins here
: 1120 6          BEGIN
: 1121 6
: 1122 6      UBI_MAP(.N) = .MAP_ENTRY;
: 1123 6      TEMP = .UBI_MAP(.N) AND MAP_MASK;
: 1124 6      IF .TEMP NEQ MAP_MASK
: 1125 6      THEN
: 1126 7          BEGIN
: 1127 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP,PRLINK=PRINT_MAP_ERR,! M7 M11
: 1128 7      P1=.N,P2=UBI_MAP(.N),P3=FMT_MAP_BIT_SA1,
: 1129 7      P4=MAP_MASK,P5=.TEMP);
: 1130 7      Test 5, Subtest 0, Error 2
: 1131 6      ! $DS_PRINTX(FMT_MAP_BIT_SA1,.N,UBI_MAP(.N),MAP_MASK,.TEMP); ! D11
: 1132 6      END;
: 1133 6      END
: 1134 5      WHILE $DS_INLOOP;
: 1135 5

```

```
; 1136 4      END;
; 1137 4
; 1138 3      END;
; 1139 3
; 1140 2      END;
; 1141 2
; 1142 2 CODECOMMENT
; 1143 2 ''
; 1144 2 'Finally, write all zeroes to each entry again. This is to leave the maps',
; 1145 2 'in a clean state.',
; 1146 2 ''
; 1147 2
; 1148 3      BEGIN
; 1149 3
; 1150 3      PFN = ALL_ZEROS;
; 1151 3      V = ALL_ZEROS;
; 1152 3      O = ALL_ZEROS;
; 1153 3      DP = ALL_ZEROS;
; 1154 3      INCR N FROM 0 TO 511 DO
; 1155 4          BEGIN
; 1156 4
; 1157 4          UBI_MAP(.N) = .MAP_ENTRY;
; 1158 4
; 1159 3      END;
; 1160 3
; 1161 2      END;
; 1162 2
; 1163 1 $DS_ENDTEST;
```

! A7
! M7

```
                                .PSECT DISPATCH,NOWRT,NOEXE,2
                                .ADDRESS P.AAM, P.AAN, P.AAO, TEST_005
00000000' 00000000' 00000000' 00000000' 00060 $: .LONG 3, 0
                                .PSECT $PLIT$,NOWRT,NOEXE,2
                                .WORD 5
74 73 65 54 20 79 72 74 6E 45 20 70 61 4D 0E 0005 00078 P.AAM: .ASCII <14>\Map Entry Test\
                                .BLKB 3
                                .LONG 0
                                .PSECT TEST_005,NOWRT,2
                                .ENTRY TEST_005, Save R2,R3,R4,R5,R6,R7,R8 ; 1028
58 0000G CF 9E 00002 MOVAB UBI_UNDER_TEST, R8
57 00000000G 9F 9E 00007 MOVAB a#DS$INLOOP, R7
56 00000000G 9F 9E 0000E MOVAB a#DS$ERRHARD, R6
                                ; Write zeros to every map entry.
52 82607FFF 8F CA 00015 BICL2 #-2107604993, MAP_ENTRY ; 1062
```

Address	Hex	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_005: Map Entry Test
TEST_005: Map Entry Test

M 14

6-Sep-1985

Fiche 1 Frame M14

Sequence 181

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 38

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(7)

FF6F

54

B9

01 000001FF

50

8F F1 000C0

E8 000BD

B1BS

R0, 5\$

ACBL

#511, #1, N, 2\$

;

; 1085

;

; Finally, write all zeroes to each entry again. This is to leave the maps in a clean state.

52 82607FFF

8F

CA 000CA

BICL2

#-2107604993, MAP_ENTRY

; 1153

50

D4 000D1

CLRL

N

; 1157

51

00 B840

DE 000D3

7\$:

MOVAL

@UBI_UNDER_TEST[N], R1

;

C1

52

D0 000D8

MOVL

MAP_ENTRY, 2048(R1)

;

EE

0800

50

9F

50

000001FF

8F

F3 000DD

AOBLEQ

#511, N, 7\$

; 1154

00000000G

00

FB 000E5

CALLS

#0, @DS\$BREAK

; 1161

01

D0 000EC

MOVL

#1, R0

; 1163

04 000EF

RET

;

; Routine Size: 240 bytes, Routine Base: TEST_005 + 0000

ZZ-ECCBA-1.4
UBI_TESTS_1_5
01-04

TEST_005: Map Entry Test
TEST_005: Map Entry Test

N 14

6-Sep-1985

Fiche 1

Frame N14

Sequence 182

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 39

6-Sep-1985 17:14:50

[LEMIEUX.ECCBA.RELEASE]ECCBA3.B32;6

(8)

; 1164 1 \$DS_ENDMOD;
; 1165 1 END
; 1166 0 ELUDOM

.PSECT \$TSTCNT,NOWRT,NOEXE, OVR,2

00000005 00000 L_TSTCNT:

.LONG 5

PSECT SUMMARY

Name	Bytes	Attributes
\$OWN\$	22	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLIT\$	144	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
DISPATCH	120	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_001	163	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_002	677	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_003	143	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_004	178	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_005	240	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$TSTCNT	4	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, OVR, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
SYS\$COMMON:[SYSLIB]STARLET.L32;3	10093	1	0	598	00:00.8
SYS\$COMMON:[SYSLIB]DIAG.L32;265	784	16	2	85	00:00.3

COMMAND QUALIFIERS

; BLISS/LIST ECCBA3.B32

; Size: 1401 code + 290 data bytes
; Run Time: 00:18.2
; Elapsed Time: 00:24.9
; Lines/CPU Min: 3837
; Lexemes/CPU-Min: 31425
; Memory Used: 216 pages

ZZ-ECCBA-1.4 TEST_005: Map Entry Test
UBI_TESTS_1_5
01-04 TEST_005: Map Entry Test

; Compilation Complete

B 15

6-Sep-1985

Fiche 1

Frame B15

Sequence 183

6-Sep-1985 17:18:43

VAX-11 Bliss-32 V4.1-746

Page 40


```

: 0001 0  MODULE UBI_TESTS_6_10 ( ! UBI diagnostic program tests 6 to 10
: 0002 0      IDENT = '01-04'
: 0003 0      ) =
: 0004 1  BEGIN
: 0005 1
: 0006 1  !
: 0007 1  ! COPYRIGHT (C) 1980
: 0008 1  ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
: 0009 1  !
: 0010 1  ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
: 0011 1  ! COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
: 0012 1  ! ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
: 0013 1  ! MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
: 0014 1  ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
: 0015 1  ! TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
: 0016 1  ! REMAIN IN DEC.
: 0017 1  !
: 0018 1  ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
: 0019 1  ! AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
: 0020 1  ! CORPORATION.
: 0021 1  !
: 0022 1  ! DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
: 0023 1  ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
: 0024 1  !
: 0025 1  !
: 0026 1  ! **
: 0027 1  ! FACILITY:      VAX-11 Diagnostic Library
: 0028 1  !
: 0029 1  ! ABSTRACT:      This module contains tests 6 to 10 of the COMET
: 0030 1  !                  Unibus Interface Diagnostic Program.
: 0031 1  !
: 0032 1  ! ENVIRONMENT:    VAX-11 Diagnostic Supervisor
: 0033 1  !
: 0034 1  ! AUTHOR:         Rand Gray, CREATION DATE: 11-Dec-78
: 0035 1  !
: 0036 1  ! MODIFIED BY:    Don Monroe : May 1979
: 0037 1  !
: 0038 1  !             0-7   May 1979
: 0039 1  !             Changed the format of the UBI MAP registers.
: 0040 1  !
: 0041 1  !             0-8   July 1979
: 0042 1  !             Modified to support the real UET module.
: 0043 1  !
: 0044 1  !             Fixed a BUG in XFER_SETUP routine. Loop that setup the maps
: 0045 1  !             would not work if MAP_NUM was not zero.
: 0046 1  !
: 0047 1  !             0-9   July 1979
: 0048 1  !             Changed referances to 'Byte Offset' to 'Byte Number'
: 0049 1  !
: 0050 1  !
: 0051 1  !             0-10  October 1979
: 0052 1  !             Added a return value to the MAP_BUFFERS routine to support
: 0053 1  !             the changes to Module 7 Revision 10.
: 0054 1  !             Added byte-offset parameter to MAP_BUFFERS routine so routine
: 0055 1  !             returns with one additional page per buffer.

```

```
; 0056 1 !
; 0057 1 !
; 0058 1 ! 0-11 October 22,1979
; 0059 1 ! Added a routine to print MAP errors. Called via PRINT LINK
; 0060 1 ! in error hard calls.
; 0061 1 !
; 0062 1 ! 0-12 December 7,1979
; 0063 1 ! Fixed bug in PROBE_ADDRESS routine in the ADAWI builtin.
; 0064 1 !
; 0065 1 ! 0-13 April 7,1980
; 0066 1 ! Changed address of unibus space so second UBI will work.
; 0067 1 !
; 0068 1 ! 0-14 April 8,1980
; 0069 1 ! Added a print general routine and converted PRINTX after
; 0070 1 ! error calls to print links.
; 0071 1 ! 1-00 June 16,1980
; 0072 1 ! Initial Release
; 0073 1 ! 1-01 July 31,1980
; 0074 1 ! Fixed bug in MAP_BUFFERS routine that would not detect
; 0075 1 ! insufficient memory for the requested number of buffers.
; 0076 1 ! 1-02 December 29,1980
; 0077 1 ! Fixed bug in MAP_BUFFERS routine. Supervisor allocates memory
; 0078 1 ! even if not enough for requested buffer size.
; 0079 1 ! 1-03 May 6, 1981
; 0080 1 ! Fixed bug in map_buffers routine.
; 0081 1 !
; 0082 1 ! Kevin Lemieux
; 0083 1 !
; 0084 1 ! 1-04 February,1985
; 0085 1 ! Fixed bug in INIT code. If two DW's were specified; after the
; 0086 1 ! first pass one of them one of them was no longer tested.
; 0087 1 ! Fixed minor bugs in TEST 29.
; 0087 1 !--
```



```

: 0088 1 !
: 0089 1 ! TABLE OF CONTENTS:
: 0090 1 !
: 0091 1 ! CPU Read/Write Test
: 0092 1 ! CMI to Unibus Address Test
: 0093 1 ! Unibus to CMI Address Test
: 0094 1 ! Data Path Select Test
: 0095 1 ! Direct Data Path DATI Test
: 0096 1 !
: 0097 1 !
: 0098 1 !
: 0099 1 ! INCLUDE FILES:
: 0100 1 !
: 0101 1 !
: 0102 1 LIBRARY 'SYS$LIBRARY:STARLET';
: 0103 1 LIBRARY 'SYS$LIBRARY:DIAG';
: 0104 1 !
: 0105 1 !
: 0106 1 ! MACROS:
: 0107 1 !
: 0108 1 !
: 0109 1 !
: 0110 1 ! EQUATED SYMBOLS:
: 0111 1 !
: 0112 1 !
: 0113 1 LITERAL
: 0114 1 ALL_ONES = %X'FFFFFFFF',
: 0115 1 ALL_ZEROS = 0,
: 0116 1 BR4 = 3,
: 0117 1 BR5 = 5,
: 0118 1 BR6 = 9,
: 0119 1 BR7 = 17,
: 0120 1 CSR_MASK = %X'E0000001',
: 0121 1
: 0122 1 DATI = 1,
: 0123 1 DATIP = 3,
: 0124 1 DATO = 5,
: 0125 1 DATOB = 7,
: 0126 1
: 0127 1 DIAG_CK_MODE = %X'C000000',
: 0128 1 DISABLE_ECC = %X'2000000',
: 0129 1 ENABLE_ECC = %X'2000000',
: 0130 1 NORMAL = %X'E000000',
: 0131 1 ERR = BIT31,
: 0132 1 MAP_MASK = %X'82607FFF',
: 0133 1 MEMORY_CONT = %X'F20000',
: 0134 1 NON_XIST_NEXUS = %X'F40000',
: 0135 1 NXM = BIT30,
: 0136 1 PURGE = BIT0,
: 0137 1 UCE = BIT29;
: 0138 1 !
: 0139 1 ! OWN STORAGE:
: 0140 1 !
: 0141 1 ! OWN
: 0142 1 BYTE_PATTERN: VECTOR[4,BYTE], ! Data patterns loaded at run time

```

! M7


```

: 0143 1 STATUS0: SIGNED WORD, ! Used by the UET status macro
: 0144 1 STATUS1, ! Used by the UBI status macro
: 0145 1 WORD_PATTERN: VECTOR[5,WORD]; ! Data patterns loaded at run time
: 0146 1
: 0147 1 !
: 0148 1 ! EXTERNAL REFERENCES:
: 0149 1 !
: 0150 1 EXTERNAL ROUTINE
: 0151 1 DECODER,
: 0152 1 ENCODER,
: 0153 1 MAP_SETUP,
: 0154 1 PRINT_GENERAL:NOVALUE,
: 0155 1 PROBE_ADDRESS,
: 0156 1 UNIBUS_SIZER;
: 0157 1
: 0158 1 EXTERNAL
: 0159 1 BUFFER_SETUP,
: 0160 1 CODEWORDS: VECTOR[9,WORD],
: 0161 1 UBI_UNDER_TEST,
: 0162 1 UNIBUS_SPACE,
: 0163 1 DECODED_WORDS: VECTOR[9],
: 0164 1 EVENT_FLAG: BITVECTOR[4],
: 0165 1 FMT_BDP-DAT1,
: 0166 1 FMT_BDP-DAT0,
: 0167 1 FMT_BDP-DAT0B,
: 0168 1 FMT_BYTE_OFF,
: 0169 1 FMT_CMI_UB,
: 0170 1 FMT_CPU_RW,
: 0171 1 FMT_DDP-DAT1,
: 0172 1 FMT_DDP-DAT0,
: 0173 1 FMT_DDP-DAT0B,
: 0174 1 FMT_DP_SEL,
: 0175 1 FMT_MAP_ADDR,
: 0176 1 ! FMT_MAP_BIT, ! D11
: 0177 1 FMT_MAP_CS,
: 0178 1 FMT_MAP-DB-SA0, ! M8
: 0179 1 FMT_MAP-DB-SA1, ! M8
: 0180 1 FMT_MAP_FAIL,
: 0181 1 FMT_MCHK,
: 0182 1 FMT-UA1,
: 0183 1 FMT-UB_CMI,
: 0184 1 FMT-UET_STAT,
: 0185 1 FMT-UB_TO, ! A7
: 0186 1 FREE_MAP: VECTOR[4,WORD],
: 0187 1 HANDLER,
: 0188 1 INTERRUPT_EXP: BYTE,
: 0189 1 INTERRUPT_OCC: BYTE,
: 0190 1 LUN: BYTE,
: 0191 1 MAP_BUFFERS,
: 0192 1 NOISY_WORDS: VECTOR[9,WORD],
: 0193 1 NUM_UBE,
: 0194 1 PRINT_CSR_ERR,
: 0195 1 PRINT_DCMF_ERR,
: 0196 1 UBIMOD, ! A7
: 0197 1 UBIMOD_BDP,

```

```

: 0198 1  UBIMOD_BYTE_OFF,
: 0199 1  UBIMOD_CMI,
: 0200 1  UBIMOD_CPU_RW,
: 0201 1  UBIMOD_CSR,
: 0202 1  UBIMOD_DDP,
: 0203 1  UBIMOD_DP_SEL,
: 0204 1  UBIMOD_MAP,
: 0205 1  UBIMOD_MAP_CS,
: 0206 1  UBIMOD_MAP_ADDR,
: 0207 1  UBIMOD_MAP_CTRL,
: 0208 1  UBIMOD-UA1,
: 0209 1  UBIMOD-UB_CMI,
: 0210 1  SCRATCH: VECTOR[128],
: 0211 1  UBE_BASE_ADDR: VECTOR[4],
: 0212 1  UBE_BUF_SIZE,
: 0213 1  ! UBE_ERROR, ! D8
: 0214 1  UBE_STAT_ERR: BITVECTOR[4],
: 0215 1  UBE0_ISR,
: 0216 1  UETMOD, ! A8
: 0217 1  UET_ISR,
: 0218 1  VALID_MAPS: BITVECTOR[512],
: 0219 1  XFER_SETUP;
: 0220 1
: 0221 1  MACRO
: 0222 1  !++
: 0223 1  ! This macro defines the address of the control and status register
: 0224 1  ! for the given buffered data path number.
: 0225 1  !--
: 0226 1  UBI_CSR(BDP) = (.UBI_UNDER_TEST + BDP*4)%;
: 0227 1
: 0228 1  !++
: 0229 1  ! This macro defines the address of a UBI map for the given map
: 0230 1  ! number.
: 0231 1  !--
: 0232 1  UBI_MAP(NUM) = (.UBI_UNDER_TEST + %X'800' + NUM*4)%;
: 0233 1
: 0234 1  !++
: 0235 1  ! These macros temporarily define the addresses of the (simulated) UET
: 0236 1  ! registers. They will be replaced for the real UET.
: 0237 1  !--
: 0238 1  MACRO ! A8
: 0239 1  UET_ADDR_REG = (.UNIBUS_SPACE OR %X'3F460')<0,16>%; ! A8 M13
: 0240 1  UET_DATA_REG = (.UNIBUS_SPACE OR %X'3F462')<0,16>%; ! A8 M13
: 0241 1  UET_DATA_REGB(N) = (.UNIBUS_SPACE OR %X'3F462' + N)<0,8>%; ! A8 M13
: 0242 1  UET_CTRL_REG = (.UNIBUS_SPACE OR %X'3F464')<0,16>%; ! A8 M13
: 0243 1
: 0244 1  !++ ! A8
: 0245 1  ! This macro is used for checking the status of the UET. ! A8
: 0246 1  !-- ! A8
: M 0247 1  UET_STATUS(CALLOUT) = ! A8
: M 0248 1  STATUS0 = .UET_CTRL_REG; ! Read the UET control register ! A8
: M 0249 1  STATUS0 = .STATUS0 AND %X'E1'; ! A8
: M 0250 1  IF .STATUS0 NEQ 0 ! A8
: M 0251 1  THEN ! A8
: M 0252 1  BEGIN ! A8

```

```

; M 0253 1          $DS_ERRHARD(UNIT=.LUN,MSGADR=CALLOUT,
; M 0254 1          PRLINK = PRINT_GENERAL,
; M 0255 1          P1 = FMT_UET_STAT,P2 = 2,
; M 0256 1          P3 = 0,P4 = .STATUS0);          ! A8
; 0257 1          END;x;                          ! A8
; 0258 1          MACRO                          ! A8
; 0259 1          !++
; 0260 1          ! This macro is used for checking the status of the UBI.
; 0261 1          !--
; M 0262 1          UBI_STATUS(BDP,CALLOUT) =
; M 0263 1          STATUS1 = .UBI_CSR(BDP) AND CSR_MASK;
; M 0264 1          IF .STATUS1 LSS 0
; M 0265 1          THEN
; M 0266 1          $DS_ERRHARD(UNIT=.LUN,
; M 0267 1          MSGADR=CALLOUT,
; M 0268 1          PRLINK=PRINT_CSR_ERR,
; 0269 1          P1=BDP,P2=0,P3=.STATUS1);x;
; 0270 1
; 0271 1          !
; 0272 1          ! Required Diagnostic Supervisor macros.
; 0273 1          !
; L 0274 1          $DS_BGNMOD(ENV=0,TEST=6);
; xPRINT:          Bliss-32 Diagnostic Macro Library version 7.0 "DIAG.L32 (57)"
; 0275 1          $DS_DSADEF;
; 0276 1          $DS_SECDEF(ALL,UBE);
; 0277 1
; 0278 1          BUILTIN
; 0279 1          ROT;

```

! This is so we can do ROTLs

TEST_006: CPU Read/Write Test

\$DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='CPU Read/Write Test');

!+
TEST DESCRIPTION:

This tests the ability of the UBI to handle CPU reads and writes to the Unibus. The UET data and address registers are written and read, checking for stuck at one or zero faults. The following patterns are used to perform this test:

WORD_PATTERN 0101010101010101
 1010101010101010
 0011001100110011
 0000111100001111
 0000000011111111

BYTE_PATTERN 01010101
 10101010
 00110011
 00001111

Subtest 4 checks that the bits in the UET control register will clear with an init.

ASSUMPTIONS:

Test 1

REGISTER

I,
N,
TEMP;

LOCAL

TEMP_ADDRESS: VECTOR[3, LONG];

! A8

CODECOMMENT

..
'Load the data patterns into the tables WORD_PATTERN and BYTE_PATTERN',
'for later use.',
..:

BEGIN

WORD_PATTERN[0] = %X'5555';
WORD_PATTERN[1] = %X'AAAA';
WORD_PATTERN[2] = %X'3333';
WORD_PATTERN[3] = %X'0F0F';
WORD_PATTERN[4] = %X'00FF';

BYTE_PATTERN[0] = %X'55';
BYTE_PATTERN[1] = %X'33';
BYTE_PATTERN[2] = %X'0F';

```

: 0335 2      END;
: 0336 2
: 0337 2      !+
: 0338 2      ! The following subtest was added in version 1-7
: 0339 2      !-
: 0340 2
: 0341 3      $DS_BGNSUB;                      ! Subtest 1 -- Word write
: 0342 3
: 0343 3      CODECOMMENT
: 0344 3      ``
: 0345 3      'Read the UET address, data, and control registers and check for a timeout',
: 0346 3      ``;
: 0347 3
: 0348 4          BEGIN
: 0349 4              TEMP_ADDRESS[0] = UET_DATA_REG;
: 0350 4              TEMP_ADDRESS[1] = UET_ADDR_REG;
: 0351 4              TEMP_ADDRESS[2] = UET_CTRL_REG;
: 0352 4
: 0353 4              INCR I FROM 0 TO 2 DO
: 0354 4                  DO                      ! Scope loop starts here
: 0355 5                      BEGIN
: 0356 5                          IF NOT PROBE_ADDRESS(.TEMP_ADDRESS[I],0,0)
: 0357 5                          THEN
: 0358 6                              BEGIN
: 0359 6                                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD,
: 0360 6                                      PRLINK = PRINT_GENERAL,
: 0361 6                                      P1 = FMT_UB_TO,P2 = 1,P3 = .TEMP_ADDRESS[I]);
: 0362 5                                  Test 6, Subtest 1, Error 1
: 0363 5                                  END;
: 0364 4                              END
: 0365 3                          WHILE $DS_INLOOP;
: 0366 3                          END;
: 0367 2                      $DS_ENDSUB;
: 0368 2
: 0369 3                      $DS_BGNSUB;                      ! Subtest 2 -- Word read/writes
: 0370 3
: 0371 3                      CODECOMMENT
: 0372 3                      ``
: 0373 3                      'Write the UET address register with each word pattern and verify it.',
: 0374 3                      ``;
: 0375 3
: 0376 4                          BEGIN
: 0377 4                              INCR N FROM 0 TO 4 DO
: 0378 4                                  BEGIN
: 0379 5                                      DO                      ! Scope loop begins here
: 0380 5                                          BEGIN
: 0381 6                                              UET_ADDR_REG = .WORD_PATTERN[N];
: 0382 6                                              TEMP = .UET_ADDR_REG;
: 0383 6                                              IF .TEMP NEQ .WORD_PATTERN[N]
: 0384 6                                              THEN
: 0385 6                                                  BEGIN
: 0386 6                                                      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD,
: 0387 6                                                          PRLINK = PRINT_GENERAL,
: 0388 6                                                          P1 = FMT_UB_TO,P2 = 1,P3 = .TEMP_ADDRESS[I]);

```

```

; P 0389 7          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CPU_RW,
; P 0390 7          PRLINK = PRINT_GENERAL,
; P 0391 7          P1 = FMT_CPU_RW,P2 = 2,
; L 0392 7          P3 = .WORD_PATTERN[.N],P4 = .TEMP);
; %PRINT:          Test 6, Subtest 2, Error 2
; 0393 6          END;
; 0394 6
; 0395 6          END
; 0396 5          WHILE $DS_INLOOP;          ! Scope loop ends here
; 0397 5
; 0398 4          END;
; 0399 4
; 0400 3          END;
; 0401 3
; 0402 3          CODECOMMENT
; 0403 3          ''
; 0404 3          'Write the UET data register with each word pattern and verify it.',
; 0405 3          ''
; 0406 3
; 0407 4          BEGIN
; 0408 4
; 0409 4          INCR N FROM 0 TO 4 DO
; 0410 5              BEGIN
; 0411 5
; 0412 5              DO          ! Scope loop begins here
; 0413 6                  BEGIN
; 0414 6
; 0415 6                  UET_DATA_REG = .WORD_PATTERN[.N];
; 0416 6                  TEMP = .UET_DATA_REG;
; 0417 6                  IF .TEMP NEQ .WORD_PATTERN[.N]
; 0418 6                  THEN
; 0419 7                      BEGIN
; P 0420 7                      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CPU_RW,
; P 0421 7                      PRLINK = PRINT_GENERAL,
; P 0422 7                      P1 = FMT_CPU_RW,P2 = 2,
; L 0423 7                      P3 = .WORD_PATTERN[.N],P4 = .TEMP);
; %PRINT:          Test 6, Subtest 2, Error 3
; 0424 6          END;
; 0425 6
; 0426 6          END
; 0427 5          WHILE $DS_INLOOP;          ! Scope loop ends here
; 0428 5
; 0429 4          END;
; 0430 4
; 0431 3          END;
; 0432 3
; 0433 2          $DS_ENDSUB;
; 0434 2
; 0435 3          $DS_BGNSUB;          ! Subtest 3, Byte reads and writes
; 0436 3          CODECOMMENT
; 0437 3          ''
; 0438 3          'Write the UET data register (first the low byte then the high byte)',
; 0439 3          'with each byte pattern and verify it.',
; 0440 3          ''
; 0441 3

```



```

: 0442 4      BEGIN
: 0443 4
: 0444 4      INCR I FROM 0 TO 1 DO          ! Byte counter          ! A8
: 0445 4      INCR N FROM 0 TO 3 DO
: 0446 5          BEGIN
: 0447 5
: 0448 5      DO                          ! Scope loop begins here
: 0449 6          BEGIN
: 0450 6
: 0451 6      UET_DATA_REG = 0;              ! Init the data register    ! A8
: 0452 6      UET_DATA_REGB(.I) = .BYTE_PATTERN[N];          ! M8
: 0453 6      TEMP = .UET_DATA_REG;
: 0454 7      IF .TEMP NEQ (.BYTE_PATTERN[N] ^ (.I * 8))      ! M8
: 0455 6      THEN
: 0456 7          BEGIN
: 0457 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CPU_RW,
: 0458 7          PRLINK = PRINT_GENERAL,
: 0459 7          P1 = FMT_CPU_RW,P2 = 2,
: 0460 7          P3 = .BYTE_PATTERN[N]*.I*8,P4 = .TEMP);
: 0461 6      xPRINT: Test 6, Subtest 3, Error 4
: 0462 6          END;
: 0463 6      END
: 0464 5      WHILE $DS_INLOOP;              ! Scope loop ends here
: 0465 5
: 0466 4      END;
: 0467 4
: 0468 3      END;
: 0469 3      $DS_ENDSUB
: 0470 3
: 0471 3      $DS_BGNSUB
: 0472 3      CODECOMMENT
: 0473 3      'Set all the bits in the UET Control Register (except NPR, PB, TO, and PE)',
: 0474 3      'check that they all set, then clear them by writing to them. Then set them',
: 0475 3      'again and check that they clear with INIT (except A1, A0, C1, and C0).',
: 0476 3      '':
: 0477 3      BEGIN
: 0478 4      $DS_SETIPL(LEVEL=XX'1F');        ! Inhibit UET interrupts
: 0479 4
: 0480 4      INCR N FROM 0 TO 4 DO
: 0481 5          BEGIN
: 0482 5      DO                          ! Scope loop starts here
: 0483 5          BEGIN
: 0484 6      UET_CTRL_REG = .WORD_PATTERN[N] AND XX'0F1E';
: 0485 6      TEMP = .UET_CTRL_REG AND XX'0F1F';
: 0486 6      IF .TEMP NEQ (.WORD_PATTERN[N] AND XX'0F1E')
: 0487 7      THEN
: 0488 6          BEGIN
: 0489 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UETMOD,
: 0490 7          PRLINK = PRINT_GENERAL,
: 0491 7          P1 = FMT_UET_STAT,P2 = 2,
: 0492 7          P3 = (.WORD_PATTERN[N] AND XX'0F1E'),
: 0493 7          P4 = .TEMP);
: 0494 7      L 0494 7
: 0495 7      xPRINT: Test 6, Subtest 4, Error 5

```

Test 6, Subtest 4, Error 6

```
00000 BYTE_PATTERN:          .BLKB      4
00004 STATUS0:              .BLKB      2
00006                  .BLKB      2
00008 STATUS1:              .BLKB      4
0000C WORD_PATTERN:          .BLKB     10
```

ZZ-ECCBA-1.4
UBI_TESTS_6_10
01-04

TEST_006: CPU Read/Write Test

TEST_006: CPU Read/Write Test

N 15

6-Sep-1985

Fiche 1 Frame N15

Sequence 195

6-Sep-1985 17:19:10

VAX-11 Bliss-32 V4.1-746

Page 12

6-Sep-1985 17:14:53

[LEMIEUX.ECCBA.RELEASE]ECCBA4.B32;6

(3)

DSA\$AL_APTMAIL= 65024
DSA\$GL_FLAGS= 65024
DSA\$GL_APTCOM= 65028
DSA\$GL_PASSES= 65032
DSA\$GL_UNITS= 65036
DSA\$GL_SECTNO= 65040
DSA\$GL_SID= 65044
DSA\$GL_MSGTYP= 65088
DSA\$GL_ERRNO= 65092
DSA\$GL_EVENT= 65096
DSA\$GL_SUBTNO= 65100
DSA\$GL_TESTNO= 65104
DSA\$GL_PASSNU= 65108
DSA\$GL_DEVLEN= 65112
DSA\$GL_DEVNAM= 65116
DSA\$GL_MSGPTR= 65128

.EXTRN DECODER, ENCODER
.EXTRN MAP_SETUP, PRINT_GENERAL
.EXTRN PROBE_ADDRESS, UNIBUS_SIZER
.EXTRN BUFFER_SETUP, CODEWORDS
.EXTRN UBI_UNDER_TEST, UNIBUS_SPACE
.EXTRN DECODED_WORDS, EVENT_FLAG
.EXTRN FMT_BDP_DATI, FMT_BDP_DATO
.EXTRN FMT_BDP_DATO, FMT_BYTE_OFF
.EXTRN FMT_CMI_UB, FMT_CPU_RW
.EXTRN FMT_DDP_DATI, FMT_DDP_DATO
.EXTRN FMT_DDP_DATO, FMT_DP_SEL
.EXTRN FMT_MAP_ADDR, FMT_MAP_CS
.EXTRN FMT_MAP_DB_SA0, FMT_MAP_DB_SA1
.EXTRN FMT_MAP_FAIL, FMT_MCHK
.EXTRN FMT-UA1, FMT_UB_CMI
.EXTRN FMT_UET_STAT, FMT_UB_TO
.EXTRN FREE_MAP, HANDLER
.EXTRN INTERRUPT_EXP, INTERRUPT_OCC
.EXTRN LUN, MAP_BUFFERS
.EXTRN NOISY_WORDS, NUM_UBE
.EXTRN PRINT_CSR_ERR, PRINT_DCOMP_ERR
.EXTRN UBIMOD, UBIMOD_BDP
.EXTRN UBIMOD_BYTE_OFF
.EXTRN UBIMOD_CMI, UBIMOD_CPU_RW
.EXTRN UBIMOD_CSR, UBIMOD_DDP
.EXTRN UBIMOD_DP_SEL, UBIMOD_MAP
.EXTRN UBIMOD_MAP_CS, UBIMOD_MAP_ADDR
.EXTRN UBIMOD_MAP_CTRL
.EXTRN UBIMOD-UA1, UBIMOD_UB_CMI
.EXTRN SCRATCH, UBE_BASE_ADDR
.EXTRN UBE_BUF_SIZE, UBE_STAT_ERR
.EXTRN UBE0_ISR, UETMOD
.EXTRN UET_ISR, VALID_MAPS
.EXTRN XFER_SETUP, DS\$BGNSUB
.EXTRN DS\$ERRHARD, DS\$INLOOP
.EXTRN DS\$ENDSUB, DS\$SETIPL
.EXTRN DS\$BREAK

.PSECT TEST_006,NOWRT,2

				OFFC	00000	.ENTRY	TEST_006, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-;	0280	
		5B	0000G	CF	9E	00002	MOVAB	UNIBUS_SPACE, R11	
		5A	00000000G	9F	9E	00007	MOVAB	a#DS\$BGNSUB, R10	
		59	0000'	CF	9E	0000E	MOVAB	WORD_PATTERN, R9	
		58	00000000G	9F	9E	00013	MOVAB	a#DS\$INLOOP, R8	
		57	00000000G	9F	9E	0001A	MOVAB	a#DS\$ERRHARD, R7	
		5E		0C	C2	00021	SUBL2	#12, SP	
; Load the data patterns into the tables WORD_PATTERN and BYTE_PATTERN									
; for later use.									
		69	AAAA5555	8F	D0	00024	MOVL	#-1431677611, WORD_PATTERN	0325
04		A9	0F0F3333	8F	D0	0002B	MOVL	#252654387, WORD_PATTERN+4	0327
08		A9	FF	8F	9B	00033	MOVZBW	#255, WORD_PATTERN+8	0329
F4		A9	3355	8F	B0	00038	MOVW	#13141, BYTE_PATTERN	0331
F6		A9		0F	9D	0003E	MOVB	#15, BYTE_PATTERN+2	0333
				01	DD	00042	PUSHL	#1	0335
				06	DD	00044	PUSHL	#6	
		6A		02	FB	00046	CALLS	#2, DS\$BGNSUB	
; Read the UET address, data, and control registers and check for a timeout									
		50		6B	D0	00049	MOVL	UNIBUS_SPACE, R0	0349
	6E	50	0003F462	8F	C9	0004C	BISL3	#259170, R0, TEMP_ADDRESS	
04	AE	50	0003F460	8F	C9	00054	BISL3	#259168, R0, TEMP_ADDRESS+4	0350
08	AE	50	0003F464	8F	C9	0005D	BISL3	#259172, R0, TEMP_ADDRESS+8	0351
				52	D4	00066	CLRL	I	0353
				7E	7C	00068	CLRQ	-(SP)	0356
			08 AE42	DD	0006A	1\$:	PUSHL	TEMP_ADDRESS[I]	
		0000G	CF	03	FB	0006E	CALLS	#3, PROBE_ADDRESS	
		20		50	E8	00073	BLBS	R0, 2\$	
				7E	7C	00076	CLRQ	-(SP)	0361
				7E	D4	00078	CLRL	-(SP)	
			0C AE42	DD	0007A		PUSHL	TEMP_ADDRESS[I]	
			01	DD	0007E		PUSHL	#1	
			0000G	CF	9F	00080	PUSHAB	FMT_UB_TO	
			0000G	CF	9F	00084	PUSHAB	PRINT_GENERAL	
			0000G	CF	9F	00088	PUSHAB	UBIMOD	
		7E	0000G	CF	9A	0008C	MOVZBL	LUN, -(SP)	
				01	DD	00091	PUSHL	#1	
		67		0A	FB	00093	CALLS	#10, DS\$ERRHARD	
		68		00	FB	00096	CALLS	#0, DS\$INLOOP	0364
		CC		50	E8	00099	BLBS	R0, 1\$	
C8		52		02	F3	0009C	AOBLEQ	#2, I, 1\$	0354
				01	DD	000A0	PUSHL	#1	0365
				06	DD	000A2	PUSHL	#6	
		00000000G	9F	02	FB	000A4	CALLS	#2, a#DS\$ENDSUB	
				02	DD	000AB	PUSHL	#2	0367
				06	DD	000AD	PUSHL	#6	
		6A		02	FB	000AF	CALLS	#2, DS\$BGNSUB	

```

;
; Write the UET address register with each word pattern and ver
; ify it.
;
51      6B 0003F460      53 D4 000B2      CLRL      N      ; 0378
      50      6943      8F C9 000B4      BISL3      #259168, UNIBUS_SPACE, R1      ; 0382
      61      50      3C 000BC      MOVZWL      WORD_PATTERN[N], R0      ; 0384
      52      61      B0 000C0      MOVW      R0, (R1)      ;
      50      52      3C 000C3      MOVZWL      (R1), TEMP      ; 0385
      1C      52      D1 000C6      CMPL      TEMP, R0      ; 0386
      7E      1C      13 000C9      BEQL      4$      ;
      05      7E      7C 000CB      CLRQ      -(SP)      ; 0392
      02      05      BB 000CD      PUSHR      #^M<R0,R2>      ;
      0000G      02      DD 000CF      PUSHL      #2      ;
      0000G      CF 9F 000D1      PUSHAB      FMT_CPU_RW      ;
      0000G      CF 9F 000D5      PUSHAB      PRINT_GENERAL      ;
      0000G      CF 9F 000D9      PUSHAB      UBIMOD_CPU_RW      ;
      7E      0000G      CF 9A 000DD      MOVZBL      LUN, -(SP)      ;
      67      02      DD 000E2      PUSHL      #2      ;
      68      0A      FB 000E4      CALLS      #10, DS$ERRHARD      ;
      C7      00      FB 000E7      4$:      CALLS      #0, DS$INLOOP      ; 0396
      53      50      E8 000EA      BLBS      R0, 3$      ;
      C3      53      04 F3 000ED      AOBLEQ      #4, N, 3$      ; 0378
;
; Write the UET data register with each word pattern and verify
; it.
;
51      6B 0003F462      53 D4 000F1      CLRL      N      ; 0409
      50      6943      8F C9 000F3      5$:      BISL3      #259170, UNIBUS_SPACE, R1      ; 0413
      61      50      3C 000FB      MOVZWL      WORD_PATTERN[N], R0      ; 0415
      52      61      B0 000FF      MOVW      R0, (R1)      ;
      50      52      3C 00102      MOVZWL      (R1), TEMP      ; 0416
      1C      52      D1 00105      CMPL      TEMP, R0      ; 0417
      7E      1C      13 00108      BEQL      6$      ;
      05      7E      7C 0010A      CLRQ      -(SP)      ; 0423
      02      05      BB 0010C      PUSHR      #^M<R0,R2>      ;
      0000G      02      DD 0010E      PUSHL      #2      ;
      0000G      CF 9F 00110      PUSHAB      FMT_CPU_RW      ;
      0000G      CF 9F 00114      PUSHAB      PRINT_GENERAL      ;
      0000G      CF 9F 00118      PUSHAB      UBIMOD_CPU_RW      ;
      7E      0000G      CF 9A 0011C      MOVZBL      LUN, -(SP)      ;
      67      03      DD 00121      PUSHL      #3      ;
      68      0A      FB 00123      CALLS      #10, DS$ERRHARD      ;
      C7      00      FB 00126      6$:      CALLS      #0, DS$INLOOP      ; 0427
      53      50      E8 00129      BLBS      R0, 5$      ;
      C3      53      04 F3 0012C      AOBLEQ      #4, N, 5$      ; 0409
      02      02      DD 00130      PUSHL      #2      ; 0431
      00000000G      9F      06 DD 00132      PUSHL      #6      ;
      02      02      FB 00134      CALLS      #2, a#DS$ENDSUB      ;
      03      06      DD 0013B      PUSHL      #3      ; 0433
      6A      06      DD 0013D      PUSHL      #6      ;
      02      02      FB 0013F      CALLS      #2, DS$BGNSUB      ;
;
; Write the UET data register (first the low byte then the high
; byte)
; with each byte pattern and verify it.

```



```

;
56      53      53      D4 00142      ; CLRL      I      ; 0444
      03      78 00144 7$:      ASHL      #3, I, R6      ; 0444
      54      D4 00148      ; CLRL      N      ;
55      6B 0003F462 8F      C9 0014A 8$:      BISL3     #259170, UNIBUS_SPACE, R5      ; 0449
      65      B4 00152      ; CLRW      (R5)      ; 0451
      50 0003F462 E3      9E 00154      ; MOVAB     259170(R3), R0      ; 0452
      50      6B      C8 0015B      ; BISL2     UNIBUS_SPACE, R0      ;
      51      F4 A944 9A 0015E      ; MOVZBL    BYTE_PATTERN[N], R1      ;
      60      51      90 00163      ; MOVB      R1, (R0)      ;
      52      65      3C 00166      ; MOVZWL    (R5), TEMP      ; 0453
50      51      56      78 00169      ; ASHL      R6, R1, R0      ; 0454
      50      52      D1 0016D      ; CMPL      TEMP, R0      ;
      24      13 00170      ; BEQL      9$      ;
      7E      7C 00172      ; CLRQ      -(SP)      ; 0460
      52      DD 00174      ; PUSHL     TEMP      ;
50      51      53      C5 00176      ; MULL3     I, R1, R0      ;
7E      50      03      78 0017A      ; ASHL      #3, R0, -(SP)      ;
      02      DD 0017E      ; PUSHL     #2      ;
      0000G    CF 9F 00180      ; PUSHAB    FMT_CPU_RW      ;
      0000G    CF 9F 00184      ; PUSHAB    PRINT_GENERAL      ;
      0000G    CF 9F 00188      ; PUSHAB    UBIMOD_CPU_RW      ;
      7E      0000G    CF 9A 0018C      ; MOVZBL    LUN, -(SP)      ;
      04      DD 00191      ; PUSHL     #4      ;
      67      0A      FB 00193      ; CALLS     #10, DS$ERRHARD      ;
      68      00      FB 00196 9$:      CALLS     #0, DS$INLOOP      ; 0464
      AE      50      E8 00199      ; BLBS      R0, 8$      ;
AA      54      03      F3 0019C      ; AOBLEQ    #3, N, 8$      ; 0445
AO      53      01      F3 001A0      ; AOBLEQ    #1, I, 7$      ;
      03      DD 001A4      ; PUSHL     #3      ; 0468
      06      DD 001A6      ; PUSHL     #6      ;
      00000000G 9F      02      FB 001A8      ; CALLS     #2, a#DS$ENDSUB      ;
      04      DD 001AF      ; PUSHL     #4      ;
      06      DD 001B1      ; PUSHL     #6      ;
      6A      02      FB 001B3      ; CALLS     #2, DS$BGNSUB      ;
;
; Set all the bits in the UET Control Register (except NPR, PB,
; TO, and PE)
; check that they all set, then clear them by writing to them.
; Then set them
; again and check that they clear with INIT (except A1, A0, C1,
; and C0).
;
      00000000G 9F      1F      DD 001B6      ; PUSHL     #31      ; 0479
      01      FB 001B8      ; CALLS     #1, a#DS$SETIPL      ;
      53      D4 001BF      ; CLRL      N      ; 0481
51      6B 0003F464 8F      C9 001C1 10$:      BISL3     #259172, UNIBUS_SPACE, R1      ; 0484
      50      6943 3C 001C9      ; MOVZWL    WORD_PATTERN[N], R0      ; 0485
      50      FFFFF0E1 8F      CA 001CD      ; BICL2     #-3871, R0      ;
      61      50      B0 001D4      ; MOVW      R0, (R1)      ;
      52      61      3C 001D7      ; MOVZWL    (R1), TEMP      ; 0486
      52      FFFFF0E0 8F      CA 001DA      ; BICL2     #-3872, TEMP      ;
      50      52      D1 001E1      ; CMPL      TEMP, R0      ; 0487
      1C      13 001E4      ; BEQL      11$      ;
      7E      7C 001E6      ; CLRQ      -(SP)      ; 0494

```


		05	BB	001E8	PUSHR	#^M<R0,R2>	:
		02	DD	001EA	PUSHL	#2	:
	0000G	CF	9F	001EC	PUSHAB	FMT_UET_STAT	:
	0000G	CF	9F	001F0	PUSHAB	PRINT_GENERAL	:
	0000G	CF	9F	001F4	PUSHAB	UETMOD	:
7E	0000G	CF	9A	001F8	MOVZBL	LUN, -(SP)	:
		05	DD	001FD	PUSHL	#5	:
67		0A	FB	001FF	CALLS	#10, DS\$ERRHARD	:
68		00	FB	00202	CALLS	#0, DS\$INLOOP	: 0497
B9		50	E8	00205	BLBS	R0, 10\$	:
53		04	F3	00208	AOBLEQ	#4, N, 10\$	: 0481
6B	0003F464	8F	C9	0020C	BISL3	#259172, UNIBUS_SPACE, R0	: 0501
60	8F1E	8F	B0	00214	MOVW	#-28898, (R0)	: 0503
52		60	3C	00219	MOVZWL	(R0), TEMP	: 0504
52	FFFFFF0E0	8F	CA	0021C	BICL2	#-3872, TEMP	:
1E		52	D1	00223	CMPL	TEMP, #30	: 0505
		1E	13	00226	BEQL	13\$	:
		7E	7C	00228	CLRQ	-(SP)	: 0511
		52	DD	0022A	PUSHL	TEMP	:
		1E	DD	0022C	PUSHL	#30	:
		02	DD	0022E	PUSHL	#2	:
	0000G	CF	9F	00230	PUSHAB	FMT_UET_STAT	:
	0000G	CF	9F	00234	PUSHAB	PRINT_GENERAL	:
	0000G	CF	9F	00238	PUSHAB	UETMOD	:
7E	0000G	CF	9A	0023C	MOVZBL	LUN, -(SP)	:
		06	DD	00241	PUSHL	#6	:
67		0A	FB	00243	CALLS	#10, DS\$ERRHARD	:
68		00	FB	00246	CALLS	#0, DS\$INLOOP	: 0514
C0		50	E8	00249	BLBS	R0, 12\$	:
6B	0003F464	8F	C9	0024C	BISL3	#259172, UNIBUS_SPACE, R0	:
60	8000	8F	B0	00254	MOVW	#-32768, (R0)	: 0515
		7E	D4	00259	CLRL	-(SP)	: 0516
00000000G	9F	01	FB	0025B	CALLS	#1, a#DS\$SETIPL	:
		04	DD	00262	PUSHL	#4	: 0517
		06	DD	00264	PUSHL	#6	:
00000000G	9F	02	FB	00266	CALLS	#2, a#DS\$ENDSUB	:
00000000G	9F	00	FB	0026D	CALLS	#0, a#DS\$BREAK	: 0518
	50	01	D0	00274	MOVL	#1, R0	: 0520
		04	00277	RET			:

; Routine Size: 632 bytes, Routine Base: TEST_006 + 0000

\$DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='CMI to Unibus Addressing Test');

!++
TEST DESCRIPTION:

This tests the ability of the UBI to correctly map a CMI address to a Unibus address in a Unibus read (DATI) transaction. Three types of fault could occur here: the first, a stuck or shorted bit in the map entry or data paths leading to the map; the second, failure to assert the PFN field of the map entry as the CMI address; and the third, a stuck or shorted bit in the byte number address path. The first type of fault was dealt with in the map entry test (Test 5). Therefore, this test deals with the second and third varieties of faults.

Subtest one deals with the case of the UBI failing to put the contents of the map PFN onto the CMI as a memory address. The UET data register is cleared with a CPU write, and a unique data pattern is written to a CMI location. The UBI and UET are set up to access that location, and a DATI is initiated. After a nominal wait, the UET data register is read. If the register contains the correct data, then this subtest is successful. However, we do an additional DATI at this time with Unibus address bit 1 changed, in order to determine if that Unibus address bit is stuck. Since the high word of the data was different from the low word, this should result in the contents of the UET data register changing.

Subtest two deals with the case of a fault in the byte number field of the address. Unique data patterns are stored in a page of memory at the following byte number addresses:

BYTE ADDRESS (binary)	DATA(hex)
000000000	0
000000100	2
000001000	3
000010000	4
000100000	5
001000000	6
010000000	7
100000000	8
111111100	A
111111000	C
111110100	D
111101100	E
111011100	F
110111100	10
101111100	11
011111100	12

The page is initialized so that each byte contains its byte number. This sequence of addresses causes a one and a zero to be floated through bits 2 through 8 of the byte number field (bits 0 and 1 are treated with separate logic). A DATI transaction is initiated to read each location in turn. Any stuck or shorted address bit will


```

: 0576 2 ! result in the wrong data being written into the UET data
: 0577 2 ! register.
: 0578 2 !
: 0579 2 ! ASSUMPTIONS:
: 0580 2 !
: 0581 2 ! Test 1-Test 6
: 0582 2 !--
: 0583 2
: 0584 2 REGISTER
: 0585 2     BUF_PFN,
: 0586 2     MAP_NUMBER,
: 0587 2     N,
: 0588 2     UBUS_ADDR: WORD,
: 0589 2     TEMP1,
: 0590 2     TEMP2: WORD,
: 0591 2     TEMP3;
: 0592 2 ! A8
: 0593 2
: 0594 2 MACRO
: 0595 2     UBUS_PF = UBUS_ADDR<9,7>x,
: 0596 2     UBUS_BO = UBUS_ADDR<0,9>x;
: 0597 2 ! Defines the Unibus page frame field
: 0598 2 ! Defines the Unibus byte number field
: 0599 2
: 0600 2 CODECOMMENT
: 0601 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
: 0602 2 'a bit map of useable map entries. In the test which follows, the map',
: 0603 2 'to be used is from the location FREE_MAP, which contains the first',
: 0604 2 'available map.',
: 0605 2 '':
: 0606 2 BEGIN
: 0607 2
: 0608 2 UNIBUS_SIZER();
: 0609 2 MAP_NUMBER = .FREE_MAP[0];
: 0610 2 END;
: 0611 2
: 0612 2 $DS_BGNSUB;
: 0613 2 ! Subtest 1
: 0614 2 CODECOMMENT
: 0615 2 '':
: 0616 2 'Clear the UET data register, and write the data pattern to use',
: 0617 2 'into the memory location SCRATCH.',
: 0618 2 '':
: 0619 2 BEGIN
: 0620 2
: 0621 2 UET_DATA_REG = 0;
: 0622 2 SCRATCH = %X'12345678';
: 0623 2
: 0624 2 END;
: 0625 2
: 0626 2 CODECOMMENT
: 0627 2 '':
: 0628 2 'Set up the UBI and UET for the transaction. Then',
: 0629 2 'initiate the DATI, wait a nominal period of time, and check the data in',
: 0630 2

```



```

: 0631 3 'the UET data register.',
: 0632 3 ''
: 0633 3
: 0634 4 BEGIN
: 0635 4
: 0636 4 DO ! Scope loop starts here
: 0637 5 BEGIN
: 0638 5
: 0639 5 CODECOMMENT
: 0640 5 ''
: 0641 5 'Set up the UET address register with the Unibus address.',
: 0642 5 'Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs of the',
: 0643 5 'UBI map number. Bits 0 to 8 contain the byte number field of',
: 0644 5 'the CMI address.',
: 0645 5 ''
: 0646 6 BEGIN
: 0647 6
: 0648 6 UBUS_PF = .MAP_NUMBER;
: 0649 6 UBUS_BO = SCRATCH;
: 0650 6 UET_ADDR_REG = .UBUS_ADDR;
: 0651 6
: 0652 5 END;
: 0653 5
: 0654 5 CODECOMMENT
: 0655 5 ''
: 0656 5 'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
: 0657 5 'UBI map number.',
: 0658 5 ''
: 0659 6 BEGIN
: 0660 6
: 0661 6 TEMP3 = .MAP_NUMBER<7,2> * 8;
: 0662 6
: 0663 5 END;
: 0664 5
: 0665 5 CODECOMMENT
: 0666 5 ''
: 0667 5 'Set up the UBI map. Put the CMI page frame number (PFN) into',
: 0668 5 'the register BUF_PFN. Then call the map setup routine with the',
: 0669 5 'parameters map number, PFN of the CMI buffer, byte offset mode',
: 0670 5 '(0 = no address offset), and data path number (0 is DDP).',
: 0671 5 ''
: 0672 6 BEGIN
: 0673 6
: 0674 6 TEMP1 = SCRATCH;
: 0675 6 BUF_PFN = .TEMP1<9,15>;
: 0676 6 MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,0);
: 0677 6
: 0678 5 END;
: 0679 5
: 0680 5 CODECOMMENT
: 0681 5 ''
: 0682 5 'Execute the DATI, wait for a nominal period of time (100 us),',
: 0683 5 'and then read the UET data register. Compare the',
: 0684 5 'contents of the UET data register (which we have stored',
: 0685 5 'in register TEMP2) with the contents of the low word of the CMI',

```

```

: 0686 5      'location SCRATCH. If they are not the same, report the error.',
: 0687 5      '':
: 0688 6      BEGIN
: 0689 6
: 0690 6      UET_CTRL_REG = .TEMP3 OR DAT1;
: 0691 6
: 0692 6      $DS_WAITUS(TIME=10);
: L 0693 7      UET_STATUS(UBIMOD_MAP_CTRL);
: xPRINT: Test 7, Subtest 1, Error 1 ! M7
: 0694 6      TEMP2 = .UET_DATA_REG;
: 0695 6      IF .TEMP2 NEQ %X'5678'
: 0696 6      THEN
: 0697 7      BEGIN
: P 0698 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP_CTRL,
: P 0699 7      PRLINK = PRINT_GENERAL,
: P 0700 7      P1 = FMT_CMI_UB,P2 = 2,
: L 0701 7      P3 = %X'5678',P4 = .TEMP2);
: xPRINT: Test 7, Subtest 1, Error 2
: 0702 6      END;
: 0703 6
: 0704 5      END;
: 0705 5
: 0706 5      END
: 0707 4      WHILE $DS_INLOOP; ! Scope loop ends here
: 0708 4
: 0709 3      END;
: 0710 3
: 0711 3      CODECOMMENT
: 0712 3      'Now change Unibus address bit 1 in the UET address register.',
: 0713 3      'initiate another DAT1, and check that the data in the UET data',
: 0714 3      'register has changed.',
: 0715 3      '':
: 0716 3
: 0717 3      BEGIN
: 0718 4
: 0719 4      UBUS_ADDR = .UBUS_ADDR + 2;
: 0720 4      UET_ADDR_REG = .UBUS_ADDR;
: 0721 4
: 0722 4      DO ! Scope loop starts here
: 0723 4      BEGIN
: 0724 5
: 0725 5      UET_CTRL_REG = DAT1 OR .TEMP3;
: 0726 5
: 0727 5      $DS_WAITUS(TIME=10);
: L 0728 5      UET_STATUS(UBIMOD_UA1); ! M7
: xPRINT: Test 7, Subtest 1, Error 3
: 0730 5      TEMP2 = .UET_DATA_REG;
: 0731 5      IF .TEMP2 NEQ %X'1234'
: 0732 5      THEN
: 0733 6      BEGIN
: P 0734 6      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_UA1,
: P 0735 6      PRLINK = PRINT_GENERAL,
: P 0736 6      P1 = FMT_UA1,P2 = 2,
: L 0737 6      P3 = %X'1234',P4 = .TEMP2);

```



```

: xPRINT:
: 0738 5
: 0739 5
: 0740 5
: 0741 4
: 0742 4
: 0743 3
: 0744 2
: 0745 2
: 0746 3
: 0747 3
: 0748 3
: 0749 3
: 0750 3
: 0751 3
: 0752 3
: 0753 3
: 0754 4
: 0755 4
: 0756 5
: 0757 5
: 0758 5
: 0759 5
: 0760 4
: 0761 4
: 0762 4
: 0763 4
: 0764 4
: 0765 4
: 0766 4
: 0767 4
: 0768 4
: 0769 4
: 0770 4
: 0771 4
: 0772 4
: 0773 4
: 0774 4
: 0775 4
: 0776 4
: 0777 4
: 0778 4
: 0779 4
: 0780 4
: 0781 4
: 0782 3
: 0783 3
: 0784 3
: 0785 3
: 0786 3
: 0787 3
: 0788 3
: 0789 3
: 0790 3
: 0791 3

Test 7, Subtest 1, Error 4
END;

END
WHILE $DS_INLOOP;          ! Scope loop ends here

END;
$DS_ENDSUB;                ! End of subtest 1
$DS_BGNSUB;                ! Subtest 2
CODECOMMENT
''
'Initialize the UET data register and store the data patterns to',
'be used into SCRATCH.',
'';

BEGIN

    BEGIN
    MAP SCRATCH: VECTOR(512,BYTE); ! A10
    INCR N FROM 0 TO 511 DO       ! A10
        SCRATCH[.N] = .N;        ! A10
    END;                          ! A10

    UET_DATA_REG = -1;

    SCRATCH[0] = 0;              ! Patterns to cause a floating one
    SCRATCH[1] = 2;              ! in the byte number address
    SCRATCH[2] = 3;
    SCRATCH[4] = 4;
    SCRATCH[8] = 5;
    SCRATCH[16] = 6;
    SCRATCH[32] = 7;
    SCRATCH[64] = 8;

    SCRATCH[127] = 10;           ! Patterns to cause a floating zero
    SCRATCH[126] = 12;           ! in the byte number address
    SCRATCH[125] = 13;
    SCRATCH[123] = 14;
    SCRATCH[119] = 15;
    SCRATCH[111] = 16;
    SCRATCH[95] = 17;
    SCRATCH[63] = 18;

    END;

CODECOMMENT
''
'Float a one through each byte number bit of the bits 2 through 8.',
'Set up a DATI and initiate it for each of the longword elements of the',
'vector SCRATCH in the sequence 0, 1, 2, 4, 8, 16, 32, 64. After the',
'DATI, check the UET data register for the correct contents. A',
'stuck or shorted byte number address bit will show up as incorrect data',
'in the UET data register.',

```



```

; 0792 3
; 0793 3
; 0794 4
; 0795 4
; 0796 4
; 0797 5
; 0798 5
; 0799 5
; 0800 6
; 0801 6
; 0802 6
; 0803 6
; 0804 6
; 0805 6
; 0806 6
; 0807 6
; 0808 6
; 0809 6
; 0810 7
; 0811 7
; 0812 7
; 0813 7
; 0814 7
; 0815 7
; 0816 7
; 0817 6
; 0818 6
; 0819 6
; 0820 6
; 0821 6
; 0822 6
; 0823 6
; 0824 7
; 0825 7
; 0826 7
; 0827 7
; 0828 6
; 0829 6
; 0830 6
; 0831 6
; 0832 6
; 0833 6
; 0834 6
; 0835 6
; 0836 6
; 0837 7
; 0838 7
; 0839 7
; 0840 7
; 0841 7
; 0842 7
; 0843 6
; 0844 6
; 0845 6
; 0846 6

'';
BEGIN
INCR M FROM 0 TO 7 DO
    BEGIN
        DO ! Scope loop starts here
            BEGIN
                CODECOMMENT
                '';
                'Set up the UET address register with the Unibus address.',
                'Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs of the',
                'UBI map number. Bits 0 to 8 contain the byte number field of',
                'the CMI address. A one is rotated through the byte number',
                'here.',
                '';
                BEGIN
                    UBUS_PF = .MAP_NUMBER;
                    N = 1 ^ (.M - 1); ! N = 2**(.M-1)
                    UBUS_BO = SCRATCH(.N);
                    UET_ADDR_REG = .UBUS_ADDR;
                END;
            CODECOMMENT
            '';
            'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
            'UBI map number.',
            '';
            BEGIN
                TEMP3 = .MAP_NUMBER<7,2> * 8;
            END;
        CODECOMMENT
        '';
        'Set up the UBI map. Put the CMI page frame number (PFN) into',
        'the register BUF_PFN. Then call the map setup routine with the',
        'parameters map number, PFN of the CMI buffer, byte offset mode',
        '(0 = no address offset), and data path number (0 is DDP).',
        '';
        BEGIN
            TEMP1 = SCRATCH;
            BUF_PFN = .TEMP1<9,15>;
            MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,0);
        END;
    CODECOMMENT
    '';

```

```

: 0847 6      'Execute the DATI, wait for a nominal period of time (100 us),',
: 0848 6      'and then read the UET data register. Compare the',
: 0849 6      'contents of the UET data register (which we have stored',
: 0850 6      'in register TEMP1) with the contents of the low word of the CMI',
: 0851 6      'location SCRATCH. If they are not the same, report the error.',
: 0852 6      '';
: 0853 7      BEGIN
: 0854 7          UET_CTRL_REG = .TEMP3 OR DATI;
: 0855 7
: 0856 7          $DS_WAITUS(TIME=10);
: 0857 7          UET_STATUS(UBIMOD_BYTE_OFF);          ! M7
L 0858 8      Test 7, Subtest 2, Error 5
: 0859 7          TEMP2 = .UET_DATA_REG;
: 0860 7          TEMP1 = .SCRATCH[N];
: 0861 7          IF .TEMP2 NEQ .TEMP1<0,16>
: 0862 7      THEN
: 0863 8          BEGIN
: 0864 8              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BYTE_OFF,
: 0865 8                  PRLINK = PRINT_GENERAL,
: 0866 8                  P1 = FMT_BYTE_OFF,P2 = 2,
: 0867 8                  P3 = .TEMP1<0,16>,P4 = .TEMP2);
: 0868 7      xPRINT: Test 7, Subtest 2, Error 6
: 0869 7          END;
: 0870 6      END;
: 0871 6      END
: 0872 6      END
: 0873 5      WHILE $DS_INLOOP;          ! Scope loop ends here
: 0874 5      END;
: 0875 4      END;
: 0876 4      END;
: 0877 3      END;
: 0878 3      CODECOMMENT
: 0879 3      ''
: 0880 3      'Now float a zero through each byte number bit of the bits 2 through 8.',
: 0881 3      'Set up a DATI and initiate it for each of the longword elements of the',
: 0882 3      'vector SCRATCH in the sequence 127, 126, 125, 123, 119, 111, 95, 63.',
: 0883 3      'After the DATI, check the UET data register for the correct',
: 0884 3      'contents. A stuck or shorted byte number address bit will show up as',
: 0885 3      'incorrect data in the UET data register.',
: 0886 3      '';
: 0887 3      BEGIN
: 0888 3      INCR M FROM 0 TO 7 DO
: 0889 4          BEGIN
: 0890 4              DO          ! Scope loop begins here
: 0891 5                  BEGIN
: 0892 5                  CODECOMMENT
: 0893 5                  ''
: 0894 6                  'Set up the UET address register with the Unibus address.',
: 0895 6                  'Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs of the',
: 0896 6
: 0897 6
: 0898 6
: 0899 6

```



```

: 0900 6      'UBI map number. Bits 0 to 8 contain the byte number field of',
: 0901 6      'the CMI address. A zero is rotated through the byte number',
: 0902 6      'here.',
: 0903 6      '':
: 0904 7      BEGIN
: 0905 7
: 0906 7      UBUS_PF = .MAP_NUMBER;
: 0907 7      N = 127 - (1 ^ (.M - 1));
: 0908 7      UBUS_BO = SCRATCH[N];
: 0909 7      UET_ADDR_REG = .UBUS_ADDR;
: 0910 7
: 0911 6      END;
: 0912 6
: 0913 6      CODECOMMENT
: 0914 6      '':
: 0915 6      'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
: 0916 6      'UBI map number.',
: 0917 6      '':
: 0918 7      BEGIN
: 0919 7
: 0920 7      TEMP3 = .MAP_NUMBER<7,2> * 8;
: 0921 7
: 0922 6      END;
: 0923 6
: 0924 6      CODECOMMENT
: 0925 6      '':
: 0926 6      'Set up the UBI map. Put the CMI page frame number (PFN) into',
: 0927 6      'the register BUF_PFN. Then call the map setup routine with the',
: 0928 6      'parameters map number, PFN of the CMI buffer, byte offset mode',
: 0929 6      '(0 = no address offset), and data path number (0 is DDP).',
: 0930 6      '':
: 0931 7      BEGIN
: 0932 7
: 0933 7      TEMP1 = SCRATCH;
: 0934 7      BUF_PFN = .TEMP1<9,15>;
: 0935 7      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,0);
: 0936 7
: 0937 6      END;
: 0938 6
: 0939 6      CODECOMMENT
: 0940 6      '':
: 0941 6      'Execute the DATI, wait for a nominal period of time (100 us),',
: 0942 6      'and then read the UET data register. Compare the',
: 0943 6      'contents of the UET data register (which we have stored',
: 0944 6      'in register TEMP1) with the contents of the low word of the CMI',
: 0945 6      'location SCRATCH. If they are not the same, report the error.',
: 0946 6      '':
: 0947 7      BEGIN
: 0948 7
: 0949 7      UET_CTRL_REG = .TEMP3 OR DATI;
: 0950 7
: 0951 7      $DS_WAITUS(TIME=10);
: L 0952 8      UET_STATUS(UBIMOD_BYTE_OFF);
: 0953 7      TEMP2 = .UET_DATA_REG;

```

Test 7, Subtest

2, Error 7

! M7


```

; 0954 7      TEMP1 = .SCRATCH(.N);
; 0955 7      IF .TEMP2 NEQ .TEMP1<0,16>
; 0956 7      THEN
; 0957 8          BEGIN
; P 0958 8          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BYTE_OFF,
; P 0959 8          PRLINK = PRINT_GENERAL,
; P 0960 8          P1 = FMT_BYTE_OFF,P2 = 2,
; L 0961 8          P3 = .TEMP1<0,16>,P4 = .TEMP2);
; xPRINT:      Test 7, Subtest 2, Error 8
; 0962 7      END;
; 0963 7
; 0964 6          END;
; 0965 6
; 0966 6          END
; 0967 5      WHILE $DS_INLOOP;          ! Scope loop ends here
; 0968 5
; 0969 4          END;
; 0970 4
; 0971 3      END;
; 0972 2      $DS_ENDSUB;
; 0973 1      $DS_ENDTEST;

```

```

; .PSECT DISPATCH,NOWRT,NOEXE,2
; 00000000' 00000000' 00000000' 00000000' 00018 $: .ADDRESS P.AAD, P.AAE, P.AAF, TEST_007
; 00000000 00000003 00028 .LONG 3, 0
; .PSECT $PLIT$,NOWRT,NOEXE,2
; 0007 0001C P.AAD: .WORD 7
; 74 73 75 62 69 6E 55 20 6F 74 20 49 4D 43 1D 0001E P.AAE: .ASCII <29>\CMI to Unibus Addressing Test\
; 74 73 65 54 20 67 6E 69 73 73 65 72 64 64 41 0002D
; 00000000 0003C P.AAF: .LONG 0
; .EXTRN DS$WAITUS
; .PSECT TEST_007,NOWRT,2
; 00FC 00000 .ENTRY TEST_007, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0521
; 5B 0000G CF 9E 00002 MOVAB UNIBUS_SPACE, R11
; 5A 0000G CF 9E 00007 MOVAB SCRATCH, R10
; ; Call the Unibus sizer routine to locate any Unibus memory. T
; his builds
; ; a bit map of useable map entries. In the test which follows,
; the map
; ; to be used is from the location FREE_MAP, which contains the
; first
; ; available map.
; 0000G CF 00 00 FB 0000C CALLS #0, UNIBUS_SIZER ; 0607
; 53 0000G CF 3C 00011 MOVZWL FREE_MAP, MAP_NUMBER ; 0608

```

```

                                01 DD 00016      PUSHL #1                ; 0610
                                07 DD 00018      PUSHL #7                ;
                                02 FB 0001A      CALLS #2, a#DS$BGNSUB    ;
00000000G 9F

```

```

; Clear the UET data register, and write the data pattern to us
; into the memory location SCRATCH.
;

```

```

50          6B 0003F462      8F C9 00021      BISL3 #259170, UNIBUS_SPACE, R0      ; 0620
          60          8F B4 00029      CLRW (R0)                ; 0622
          6A 12345678      8F D0 0002B      MOVL #305419896, SCRATCH      ; 0623

```

```

; Set up the UBI and UET for the transaction. Then
; initiate the DATI, wait a nominal period of time, and check t
; he data in
; the UET data register.
;

```

```

; Set up the UET address register with the Unibus address.
; Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs of th
; e
; UBI map number. Bits 0 to 8 contain the byte number field of
; the CMI address.
;

```

```

55          07          09          53 F0 00032      1$: INSV MAP_NUMBER, #9, #7, UBUS_ADDR      ; 0648
          50          6A 9E 00037      MOVAB SCRATCH, R0                ; 0649
55          09          00          50 F0 0003A      INSV R0, #0, #9, UBUS_ADDR      ;
          50          6B 0003F460      8F C9 0003F      BISL3 #259168, UNIBUS_SPACE, R0      ;
          60          55 B0 00047      MOVW UBUS_ADDR, (R0)            ; 0650

```

```

; Write the 2 MSBs of the Unibus address with the 2 MSBs of the
; UBI map number.
;

```

```

58          53          02          07 EF 0004A      EXTZV #7, #2, MAP_NUMBER, TEMP3      ; 0661
          58          08 C4 0004F      MULL2 #8, TEMP3                ;

```

```

; Set up the UBI map. Put the CMI page frame number (PFN) into
; the register BUF_PFN. Then call the map setup routine with t
; he
; parameters map number, PFN of the CMI buffer, byte offset mod
; e
; (0 = no address offset), and data path number (0 is DDP).
;

```

```

52          56          56          6A 9E 00052      MOVAB SCRATCH, TEMP1                ; 0674
          0F          09 EF 00055      EXTZV #9, #15, TEMP1, BUF_PFN      ; 0675
          7E 7C 0005A      CLRQ -(SP)                ; 0676
          52 DD 0005C      PUSHL BUF_PFN                ;
          53 DD 0005E      PUSHL MAP_NUMBER              ;
          04 FB 00060      CALLS #4, MAP_SETUP            ;
0000G CF

```

```

; Execute the DATI, wait for a nominal period of time (100 us),
; and then read the UET data register. Compare the
; contents of the UET data register (which we have stored
; in register TEMP2) with the contents of the low word of the C
; MI

```


; location SCRATCH. If they are not the same, report the error

```

50      6B 0003F464 8F C9 00065      BISL3 #259172, UNIBUS_SPACE, R0      : 0688
60      58      01 A9 0006D      BISW3 #1, TEMP3, (R0)      : 0690
      7E      0A 7D 00071      MOVQ #10, -(SP)      : 0692
      9F      02 FB 00074      CALLS #2, a#DS$WAITUS      :
50      6B 0003F464 8F C9 0007B      BISL3 #259172, UNIBUS_SPACE, R0      : 0693
      CF      60 B0 00083      MOVW (R0), STATUS0      :
      0000' CF      FF1E 8F AA 00088      BICW2 #65310, STATUS0      :
      0000' 50      0000' CF 32 0008F      CVTWL STATUS0, R0      :
      21 13 00094      BEQL 2$      :
      7E 7C 00096      CLRQ -(SP)      :
      50 DD 00098      PUSHL R0      :
      7E      02 7D 0009A      MOVQ #2, -(SP)      :
      0000G CF 9F 0009D      PUSHAB FMT_UET_STAT      :
      0000G CF 9F 000A1      PUSHAB PRINT_GENERAL      :
      0000G CF 9F 000A5      PUSHAB UBIMOD_MAP_CTRL      :
      0000G 7E      CF 9A 000A9      MOVZBL LUN, -(SP)      :
      01 DD 000AE      PUSHL #1      :
      9F      0A FB 000B0      CALLS #10, a#DS$ERRHARD      :
50      6B 0003F462 8F C9 000B7 2$: BISL3 #259170, UNIBUS_SPACE, R0      : 0694
      57      60 B0 000BF      MOVW (R0), TEMP2      :
      8F      57 B1 000C2      CMPW TEMP2, #22136      : 0695
      26 13 000C7      BEQL 3$      :
      7E 7C 000C9      CLRQ -(SP)      : 0701
      57 3C 000CB      MOVZWL TEMP2, -(SP)      :
      7E      8F 3C 000CE      MOVZWL #22136, -(SP)      :
      02 DD 000D3      PUSHL #2      :
      0000G CF 9F 000D5      PUSHAB FMT_CMI_UB      :
      0000G CF 9F 000D9      PUSHAB PRINT_GENERAL      :
      0000G CF 9F 000DD      PUSHAB UBIMOD_MAP_CTRL      :
      0000G 7E      CF 9A 000E1      MOVZBL LUN, -(SP)      :
      02 DD 000E6      PUSHL #2      :
      00000000G 9F      0A FB 000E8      CALLS #10, a#DS$ERRHARD      :
      00000000G 9F      00 FB 000EF 3$: CALLS #0, a#DS$INLOOP      : 0707
      03      50 E9 000F6      BLBC R0, 4$      :
      FF36 31 000F9      BRW 1$      :

```

; Now change Unibus address bit 1 in the UET address register,
; initiate another DATI, and check that the data in the UET dat
a
; register has changed.

```

50      55      02 A0 000FC 4$: ADDW2 #2, UBUS_ADDR      : 0720
      6B 0003F460 8F C9 000FF      BISL3 #259168, UNIBUS_SPACE, R0      :
      60      55 B0 00107      MOVW UBUS_ADDR, (R0)      : 0721
54      58      01 C9 0010A      BISL3 #1, TEMP3, R4      : 0726
50      6B 0003F464 8F C9 0010E 5$: BISL3 #259172, UNIBUS_SPACE, R0      : 0724
      60      54 B0 00116      MOVW R4, (R0)      : 0726
      7E      0A 7D 00119      MOVQ #10, -(SP)      : 0728
      00000000G 9F      02 FB 0011C      CALLS #2, a#DS$WAITUS      :
50      6B 0003F464 8F C9 00123      BISL3 #259172, UNIBUS_SPACE, R0      : 0729
      0000' CF      60 B0 0012B      MOVW (R0), STATUS0      :
      0000' CF      FF1E 8F AA 00130      BICW2 #65310, STATUS0      :

```



```

50      0000'  CF  32 00137      CVTL      STATUS0, R0      ;
          21 13 0013C      BEQL      6$      ;
          7E 7C 0013E      CLRQ      -(SP)      ;
          50 DD 00140      PUSHL     R0      ;
7E      02 7D 00142      MOVQ      #2, -(SP)      ;
          0000G CF  9F 00145      PUSHAB  FMT_UET_STAT      ;
          0000G CF  9F 00149      PUSHAB  PRINT_GENERAL      ;
          0000G CF  9F 0014D      PUSHAB  UBIMOD_UA1      ;
          7E 0000G CF  9A 00151      MOVZBL LUN, -(SP)      ;
          03 DD 00156      PUSHL     #3      ;
50      00000000G 9F  0A FB 00158      CALLS   #10, a#DS$ERRHARD      ;
6B      0003F462 8F  C9 0015F 6$:  BISL3   #259170, UNIBUS_SPACE, R0      ; 0730
57      60 B0 00167      MOVW      (R0), TEMP2      ;
1234    8F  57 B1 0016A      CMPW      TEMP2, #4660      ; 0731
          26 13 0016F      BEQL      7$      ;
          7E 7C 00171      CLRQ      -(SP)      ; 0737
          7E 57 3C 00173      MOVZWL  TEMP2, -(SP)      ;
          7E 1234 8F  3C 00176      MOVZWL  #4660, -(SP)      ;
          02 DD 0017B      PUSHL     #2      ;
          0000G CF  9F 0017D      PUSHAB  FMT_UA1      ;
          0000G CF  9F 00181      PUSHAB  PRINT_GENERAL      ;
          0000G CF  9F 00185      PUSHAB  UBIMOD_UA1      ;
          7E 0000G CF  9A 00189      MOVZBL LUN, -(SP)      ;
          04 DD 0018E      PUSHL     #4      ;
          00000000G 9F  0A FB 00190      CALLS   #10, a#DS$ERRHARD      ;
          00000000G 9F  00 FB 00197 7$:  CALLS   #0, a#DS$INLOOP      ; 0741
          03      50 E9 0019E      BLBC     R0, 8$      ;
          FF6A 31 001A1      BRW       5$      ;
          01 DD 001A4 8$:  PUSHL     #1      ; 0743
          07 DD 001A6      PUSHL     #7      ;
          00000000G 9F  02 FB 001A8      CALLS   #2, a#DS$ENDSUB      ;
          02 DD 001AF      PUSHL     #2      ; 0744
          07 DD 001B1      PUSHL     #7      ;
          00000000G 9F  02 FB 001B3      CALLS   #2, a#DS$BGNSUB      ;
;
; Initialize the UET data register and store the data patterns
; to
; be used into SCRATCH.
;
          50 D4 001BA      CLRL      N      ; 0758
          50 90 001BC 9$:  MOVVB     N, SCRATCH[N]      ; 0759
          8F F3 001C0      AOBLEQ    #511, N, 9$      ;
          8F C9 001C8      BISL3     #259170, UNIBUS_SPACE, R0      ; 0760
          01 AE 001D0      MNEGW     #1, (R0)      ; 0762
          6A D4 001D3      CLRL      SCRATCH      ; 0764
          02 D0 001D5      MOVL      #2, SCRATCH+4      ; 0765
          03 D0 001D9      MOVL      #3, SCRATCH+8      ; 0766
          04 D0 001DD      MOVL      #4, SCRATCH+16      ; 0767
          05 D0 001E1      MOVL      #5, SCRATCH+32      ; 0768
          06 D0 001E5      MOVL      #6, SCRATCH+64      ; 0769
          07 D0 001E9      MOVL      #7, SCRATCH+128      ; 0770
          08 D0 001EE      MOVL      #8, SCRATCH+256      ; 0771
          0A D0 001F3      MOVL      #10, SCRATCH+508      ; 0773
          0C D0 001F8      MOVL      #12, SCRATCH+504      ; 0774
          0D D0 001FD      MOVL      #13, SCRATCH+500      ; 0775

```

01EC	CA	0E	D0	00202	MOVL	#14, SCRATCH+492	; 0776
01DC	CA	0F	D0	00207	MOVL	#15, SCRATCH+476	; 0777
01BC	CA	10	D0	0020C	MOVL	#16, SCRATCH+444	; 0778
017C	CA	11	D0	00211	MOVL	#17, SCRATCH+380	; 0779
00FC	CA	12	D0	00216	MOVL	#18, SCRATCH+252	; 0780

; Float a one through each byte number bit of the bits 2 through 8.
; Set up a DATI and initiate it for each of the longword elements of the
; vector SCRATCH in the sequence 0, 1, 2, 4, 8, 16, 32, 64. After the
; DATI, check the UET data register for the correct contents.
; stuck or shorted byte number address bit will show up as incorrect data
; in the UET data register.

59	D4	0021B	CLRL	M		; 0796
----	----	-------	------	---	--	--------

; Set up the UET address register with the Unibus address.
; Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs of the
; UBI map number. Bits 0 to 8 contain the byte number field of
; the CMI address. A one is rotated through the byte number
; here.

55	07	09	53	F0	0021D	10\$: INSV	MAP_NUMBER, #9, #7, UBUS_ADDR	; 0812	
		50	FF	A9	9E	00222	MOVAB	-1(R9), R0	; 0813
	54	01	50	78	00226	ASHL	R0, #1, N	; 0814	
		50	6A44	DE	0022A	MOVAL	SCRATCH[N], R0	; 0815	
55	09	00	50	F0	0022E	INSV	R0, #0, #9, UBUS_ADDR		
	50	6B	0003F460	8F	C9	00233	BISL3	#259168, UNIBUS_SPACE, R0	
		60		55	B0	0023B	MOVW	UBUS_ADDR, (R0)	

; Write the 2 MSBs of the Unibus address with the 2 MSBs of the
; UBI map number.

58	53	02	07	EF	0023E	EXTZV	#7, #2, MAP_NUMBER, TEMP3	; 0826
		58	08	C4	00243	MULL2	#8, TEMP3	

; Set up the UBI map. Put the CMI page frame number (PFN) into
; the register BUF_PFN. Then call the map setup routine with the
; parameters map number, PFN of the CMI buffer, byte offset mode
; (0 = no address offset), and data path number (0 is DDP).

52	56	56	6A	9E	00246	MOVAB	SCRATCH, TEMP1	; 0839	
		0F	09	EF	00249	EXTZV	#9, #15, TEMP1, BUF_PFN	; 0840	
			7E	7C	0024E	CLRQ	-(SP)	; 0841	
			52	DD	00250	PUSHL	BUF_PFN		
			53	DD	00252	PUSHL	MAP_NUMBER		
		0000G	CF	04	FB	00254	CALLS	#4, MAP_SETUP	

; Execute the DATI, wait for a nominal period of time (100 us),
; and then read the UET data register. Compare the
; contents of the UET data register (which we have stored
; in register TEMP1) with the contents of the low word of the C
MI
; location SCRATCH. If they are not the same, report the error

50	6B	0003F464	8F	C9	00259	BISL3	#259172, UNIBUS_SPACE, R0	; 0853
60	58		01	A9	00261	BISW3	#1, TEMP3, (R0)	; 0855
	7E		0A	7D	00265	MOVQ	#10, -(SP)	; 0857
	9F	00000000G	02	FB	00268	CALLS	#2, a#DS\$WAITUS	; 0858
50	6B	0003F464	8F	C9	0026F	BISL3	#259172, UNIBUS_SPACE, R0	; 0858
	CF	0000'	60	B0	00277	MOVW	(R0), STATUS0	; 0859
	CF	0000'	8F	AA	0027C	BICW2	#65310, STATUS0	; 0860
	50	0000'	CF	32	00283	CVTWL	STATUS0, R0	; 0861
			21	13	00288	BEQL	11\$	; 0867
			7E	7C	0028A	CLRQ	-(SP)	; 0873
			50	DD	0028C	PUSHL	R0	; 0796
	7E		02	7D	0028E	MOVQ	#2, -(SP)	; 0853
		0000G	CF	9F	00291	PUSHAB	FMT UET_STAT	; 0855
		0000G	CF	9F	00295	PUSHAB	PRINT_GENERAL	; 0857
		0000G	CF	9F	00299	PUSHAB	UBIMOD_BYTE_OFF	; 0858
	7E	0000G	CF	9A	0029D	MOVZBL	LUN, -(SP)	; 0859
			05	DD	002A2	PUSHL	#5	; 0860
50	9F	00000000G	0A	FB	002A4	CALLS	#10, a#DS\$ERRHARD	; 0861
	6B	0003F462	8F	C9	002AB	BISL3	#259170, UNIBUS_SPACE, R0	; 0867
	57		60	B0	002B3	MOVW	(R0), TEMP2	; 0873
	56		6A44	D0	002B6	MOVL	SCRATCH[N], TEMP1	; 0796
	56		57	B1	002BA	CMPW	TEMP2, TEMP1	; 0853
			24	13	002BD	BEQL	12\$	; 0855
			7E	7C	002BF	CLRQ	-(SP)	; 0857
	7E		57	3C	002C1	MOVZWL	TEMP2, -(SP)	; 0858
	7E		56	3C	002C4	MOVZWL	TEMP1, -(SP)	; 0859
			02	DD	002C7	PUSHL	#2	; 0860
		0000G	CF	9F	002C9	PUSHAB	FMT_BYTE_OFF	; 0861
		0000G	CF	9F	002CD	PUSHAB	PRINT_GENERAL	; 0867
		0000G	CF	9F	002D1	PUSHAB	UBIMOD_BYTE_OFF	; 0873
	7E	0000G	CF	9A	002D5	MOVZBL	LUN, -(SP)	; 0796
			06	DD	002DA	PUSHL	#6	; 0853
	9F	00000000G	0A	FB	002DC	CALLS	#10, a#DS\$ERRHARD	; 0855
	9F	00000000G	00	FB	002E3	CALLS	#0, a#DS\$INLOOP	; 0857
	03		50	E9	002EA	BLBC	R0, 14\$	; 0858
			FF2D	31	002ED	BRW	10\$	; 0859
F9	59		07	F3	002F0	AOBLEQ	#7, M, 13\$	; 0860

; Now float a zero through each byte number bit of the bits 2 t
hrough 8.
; Set up a DATI and initiate it for each of the longword elemen
ts of the
; vector SCRATCH in the sequence 127, 126, 125, 123, 119, 111,
95, 63.
; After the DATI, check the UET data register for the correct
; contents. A stuck or shorted byte number address bit will sh
ow up as


```

; incorrect data in the UET data register.
;
59 D4 002F4 CLRL M ; 0890
;
; Set up the UET address register with the Unibus address.
; Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs of the
; UBI map number. Bits 0 to 8 contain the byte number field of
; the CMI address. A zero is rotated through the byte number
; here.
;
55 07 09 53 F0 002F6 15$: INSV MAP_NUMBER, #9, #7, UBUS_ADDR ; 0906
50 50 FF A9 9E 002FB MOVAB -1(R9), R0 ; 0907
54 0000007F 01 50 78 002FF ASHL R0, #1, R0 ;
50 8F 50 C3 00303 SUBL3 R0, #127, N ;
55 09 50 6A44 DE 0030B MOVAL SCRATCH[N], R0 ; 0908
50 00 50 F0 0030F INSV R0, #0, #9, UBUS_ADDR ;
50 6B 0003F460 8F C9 00314 BISL3 #259168, UNIBUS_SPACE, R0 ;
60 55 B0 0031C MOVW UBUS_ADDR, (R0) ; 0909
;
; Write the 2 MSBs of the Unibus address with the 2 MSBs of the
; UBI map number.
;
58 53 02 07 EF 0031F EXTZV #7, #2, MAP_NUMBER, TEMP3 ; 0920
58 58 08 C4 0032A MULL2 #8, TEMP3 ;
;
; Set up the UBI map. Put the CMI page frame number (PFN) into
; the register BUF_PFN. Then call the map setup routine with the
; parameters map number, PFN of the CMI buffer, byte offset mod
; e
; (0 = no address offset), and data path number (0 is DDP).
;
52 56 56 6A 9E 00327 MOVAB SCRATCH, TEMP1 ; 0933
50 56 0F 09 EF 0032A EXTZV #9, #15, TEMP1, BUF_PFN ; 0934
7E 7C 0032F CLRQ -(SP) ; 0935
52 DD 00331 PUSHL BUF_PFN ;
53 DD 00333 PUSHL MAP_NUMBER ;
0000G CF 04 FB 00335 CALLS #4, MAP_SETUP ;
;
; Execute the DATI, wait for a nominal period of time (100 us),
; and then read the UET data register. Compare the
; contents of the UET data register (which we have stored
; in register TEMP1) with the contents of the low word of the C
; MI
; location SCRATCH. If they are not the same, report the error
;
50 6B 0003F464 8F C9 0033A BISL3 #259172, UNIBUS_SPACE, R0 ; 0947
60 58 01 A9 00342 BISW3 #1, TEMP3, (R0) ; 0949
7E 0A 7D 00346 MOVQ #10, -(SP) ; 0951
00000000G 9F 02 FB 00349 CALLS #2, @DS$WAITUS ;
50 6B 0003F464 8F C9 00350 BISL3 #259172, UNIBUS_SPACE, R0 ; 0952
0000' CF 60 B0 00358 MOVW (R0), STATUS0 ;
0000' CF FF1E 8F AA 0035D BICW2 #65310, STATUS0 ;

```

50	0000'	CF	32	00364	CVTL	STATUS0, R0	:
		21	13	00369	BEQL	16\$	:
		7E	7C	0036B	CLRQ	-(SP)	:
		50	DD	0036D	PUSHL	R0	:
7E		02	7D	0036F	MOVQ	#2, -(SP)	:
	0000G	CF	9F	00372	PUSHAB	FMT_UET_STAT	:
	0000G	CF	9F	00376	PUSHAB	PRINT_GENERAL	:
	0000G	CF	9F	0037A	PUSHAB	UBIMOD_BYTE_OFF	:
7E	0000G	CF	9A	0037E	MOVZBL	LUN, -(SP)	:
		07	DD	00383	PUSHL	#7	:
50	00000000G	9F	0A	FB	CALLS	#10, a#DS\$ERRHARD	:
		6B	0003F462	8F	C9	0038C	16\$:
		57		60	B0	00394	:
		56		6A44	D0	00397	:
		56		57	B1	0039B	:
				24	13	0039E	:
				7E	7C	003A0	:
7E				57	3C	003A2	:
7E				56	3C	003A5	:
				02	DD	003A8	:
		0000G	CF	9F	003AA	PUSHAB	FMT_BYTE_OFF
		0000G	CF	9F	003AE	PUSHAB	PRINT_GENERAL
		0000G	CF	9F	003B2	PUSHAB	UBIMOD_BYTE_OFF
7E		0000G	CF	9A	003B6	MOVZBL	LUN, -(SP)
				08	DD	003BB	:
	00000000G	9F	0A	FB	CALLS	#10, a#DS\$ERRHARD	:
	00000000G	9F	00	FB	CALLS	#0, a#DS\$INLOOP	:
		03		50	E9	003CB	17\$:
				FF25	31	003CE	18\$:
F9		59		07	F3	003D1	19\$:
				02	DD	003D5	:
				07	DD	003D7	:
	00000000G	9F	02	FB	CALLS	#2, a#DS\$ENDSUB	:
	00000000G	9F	00	FB	CALLS	#0, a#DS\$BREAK	:
		50	01	D0	003E7	MOVL	#1, R0
			04	003EA	RET		:

; Routine Size: 1003 bytes, Routine Base: TEST_007 + 0000


```
; 0974 2 $DS_BGNTTEST (SECTION=(DEFAULT,ALL),TEXT='Unibus to CMI Addressing Test');
; 0975 2
; 0976 2 !++
; 0977 2 ! TEST DESCRIPTION:
; 0978 2 !
; 0979 2 ! This tests the ability of the UBI to correctly select a map from the
; 0980 2 ! page frame of the Unibus address. The type of fault which could occur
; 0981 2 ! here is a stuck or shorted Unibus address bit, which would cause the
; 0982 2 ! wrong map entry to be selected. In order to detect this type of
; 0983 2 ! fault, the following steps are executed:
; 0984 2 !
; 0985 2 ! 1. Clear the UET data register.
; 0986 2 ! 2. Invalidate all maps but one.
; 0987 2 ! 3. Write a unique data pattern into a memory location.
; 0988 2 ! 4. Set up and execute a DATI from that location.
; 0989 2 ! 5. Verify that UET data register has correct data.
; 0990 2 ! 6. Do steps 1 to 5 for every useable map of the set 0, 1, 2,
; 0991 2 ! 4, 16, 32, 64, 128, and 256 (this causes a one to be
; 0992 2 ! floated across each page frame bit of the Unibus address.
; 0993 2 !
; 0994 2 ! The byte number field need not be explicitly tested, as it was tested
; 0995 2 ! in the previous test.
; 0996 2 !
; 0997 2 ! The direct data path is used for this test. Note that the presence of
; 0998 2 ! Unibus memory will preclude testing some subset of the Unibus address
; 0999 2 ! page frame bits.
; 1000 2 !
; 1001 2 ! ASSUMPTIONS:
; 1002 2 !
; 1003 2 ! Test 1-Test 7
; 1004 2 !--
; 1005 2
; 1006 2 REGISTER
; 1007 2 BUF_PFN,
; 1008 2 MAP_ENTRY,
; 1009 2 M: WORD,
; 1010 2 N,
; 1011 2 UBUS_ADDR: WORD,
; 1012 2 TEMP1,
; 1013 2 TEMP2: WORD,
; 1014 2 TEMP3;
; 1015 2
; 1016 2 MACRO
; 1017 2 V_BIT = MAP_ENTRY<31,1>%, ! Defines the UBI map valid bit field ! M7
; 1018 2 UBUS_PF = UBUS_ADDR<9,7>%, ! Defines the Unibus page frame field
; 1019 2 UBUS_BO = UBUS_ADDR<0,9>%; ! Defines the Unibus byte number field
; 1020 2
; 1021 2 CODECOMMENT
; 1022 2 ''
; 1023 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 1024 2 'a bit map of useable map entries. In the test which follows, the maps',
; 1025 2 'to be used are checked for validity against this bit map.',
; 1026 2 '';
; 1027 2
; 1028 3 BEGIN
```



```

1029 3      UNIBUS_SIZER();
1030 3
1031 3
1032 2      END;
1033 2
1034 2 CODECOMMENT
1035 2 ''
1036 2 'Invalidate all map entries and initialize the UET data register.',
1037 2 ''
1038 2
1039 3      BEGIN
1040 3
1041 3      UET_DATA_REG = -1;
1042 3      V_BIT = 0;
1043 3      INCR N FROM 0 TO 511 DO
1044 4          BEGIN
1045 4
1046 4              UBI_MAP(.N) = .MAP_ENTRY;
1047 4
1048 3          END;
1049 3
1050 2      END;
1051 2
1052 2 CODECOMMENT
1053 2 ''
1054 2 'Write a unique data pattern to a memory location, set up and execute a',
1055 2 'DATI from that location, and check that the UET data register',
1056 2 'contains the correct data. If it does not, then the correct map has',
1057 2 'not been addressed properly due to a stuck or shorted Unibus page frame',
1058 2 'bit. This sequence is repeated for each map entry of the set of map',
1059 2 'entries 0, 1, 2, 4, 8, 16, 32, 64, 128, and 256.',
1060 2 ''
1061 2
1062 3      BEGIN
1063 3
1064 3      INCR M FROM 0 TO 9 DO
1065 4          BEGIN
1066 4
1067 4              CODECOMMENT
1068 4              ''
1069 4              'Select the map number from the set 0, 1, 2, 4, 8, 16, 32, 64, 128,',
1070 4              'and 256. Check if the map selected is valid by checking the valid',
1071 4              'map bitvector. If the map is useable, go ahead with the test.',
1072 4              ''
1073 5              BEGIN
1074 5
1075 5                  N = 1 ^ (.M - 1);
1076 5                  IF .VALID_MAPS[.N]
1077 5                      THEN
1078 6                      BEGIN
1079 6
1080 6                          DO                      ! Scope loop begins here
1081 7                          BEGIN
1082 7
1083 7                          CODECOMMENT

```

```

; 1084 7
; 1085 7
; 1086 7
; 1087 7
; 1088 7
; 1089 7
; 1090 8
; 1091 8
; 1092 8
; 1093 8
; 1094 8
; 1095 8
; 1096 7
; 1097 7
; 1098 7
; 1099 7
; 1100 7
; 1101 7
; 1102 7
; 1103 8
; 1104 8
; 1105 8
; 1106 8
; 1107 7
; 1108 7
; 1109 7
; 1110 7
; 1111 7
; 1112 7
; 1113 7
; 1114 7
; 1115 7
; 1116 7
; 1117 7
; 1118 8
; 1119 8
; 1120 8
; 1121 8
; 1122 8
; 1123 8
; 1124 8
; 1125 7
; 1126 7
; 1127 7
; 1128 7
; 1129 7
; 1130 7
; 1131 7
; 1132 7
; 1133 7
; 1134 7
; 1135 7
; 1136 8
; 1137 8
; 1138 8

      ''
      'Set up the UET address register with the Unibus',
      'address. Bits 9 to 15 (the Unibus page frame) contain',
      'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
      'the byte number field of the CMI address.',
      ''
      BEGIN
          UBUS_PF = .N;
          UBUS_BO = SCRATCH;
          UET_ADDR_REG = .UBUS_ADDR;
      END;

CODECOMMENT
''
'Write the two MSBs of the Unibus address, which is the',
'two MSBs of the map number.',
''
      BEGIN
          TEMP3 = .N<7,2> * 8;
      END;

CODECOMMENT
''
'Set up the UBI map. Put the CMI page frame number',
'(PFN) into the register BUF_PFN. Then call the map',
'setup routine with the parameters map number, PFN of',
'the CMI buffer, byte offset mode (0 = no address',
'offset), and data path number (0 is DDP). Also, set up',
'the CMI source location.',
''
      BEGIN
          TEMP1 = SCRATCH;
          BUF_PFN = .TEMP1<9,15>;
          MAP_SETUP(.N,.BUF_PFN,0,0);
          SCRATCH = .M;
      END;

CODECOMMENT
''
'Execute the DATI, wait for a nominal period of time',
'(100 us), and then read the UET data register.',
'Compare the contents of the UET data register',
'(which we have stored in register TEMP2) with the',
'contents of the low word of the CMI location SCRATCH.',
'If they are not the same, report the error.',
''
      BEGIN
          UET_CTRL_REG = .TEMP3 OR DATI;

```

```

; 1139 8
; 1140 8
; L 1141 9
; XPRINT:
; 1142 8
; 1143 8
; 1144 8
; 1145 9
; P 1146 9
; P 1147 9
; P 1148 9
; L 1149 9
; XPRINT:
; 1150 8
; 1151 8
; 1152 8
; 1153 7
; 1154 7
; 1155 7
; 1156 6
; 1157 6
; 1158 5
; 1159 5
; 1160 4
; 1161 4
; 1162 3
; 1163 3
; 1164 2
; 1165 2
; 1166 1

$DS_WAITUS(TIME=10);
UET_STATUS(UBIMOD_UB_CMI); ! M7
Test 8, Subtest 0, Error 1
TEMP2 = .UET_DATA_REG;
IF .TEMP2 NEQ .M
THEN
BEGIN
$DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_UB_CMI,
PRLINK = PRINT_GENERAL,
P1 = FMT_UB_CMI,P2 = 2,
P3 = .M,P4 = .TEMP2);
Test 8, Subtest 0, Error 2
END;
UBI_MAP(.N) = .MAP_ENTRY; ! Invalidate the map
END;
END
WHILE $DS_INLOOP; ! Scope loop ends here
END;
END;
END;
END;
$DS_ENDTEST;

```

```

.PSECT DISPATCH,NOWRT,NOEXE,2
00000000' 00000000' 00000000' 00000000' 00030 $: .ADDRESS P.AAG, P.AAH, P.AAI, TEST_008
00000000 00000003 00040 .LONG 3, 0
.PSECT $PLIT$,NOWRT,NOEXE,2
0008 00040 P.AAG: .WORD 8
74 49 4D 43 20 6F 74 20 73 75 62 69 6E 55 1D 00042 P.AAH: .ASCII <29>\Unibus to CMI Addressing Test\
74 73 65 54 20 67 6E 69 73 73 65 72 64 64 41 00051
00000000 00060 P.AAI: .LONG 0
.PSECT TEST_008,NOWRT,2
OFFC 00000 .ENTRY TEST_008, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0974
5B 00000000G 9F 9E 00002 MOVAB a#DS$ERRHARD, R11
5A 0000G CF 9E 00009 MOVAB UNIBUS_SPACE, R10
; Call the Unibus sizer routine to locate any Unibus memory. T

```


:

1.

2

2

1.

1.

;

1

1

53

55

55

50		6A	0003F462	8F	C9	00013
		60		01	AE	0001B
01		1F		00	F0	0001E
				50	D4	00023
		51	0000GDF40	40	DE	00025
	0800	C1		53	D0	0002B
ED		50	000001FF	8F	F3	00030

```

54      50      FF      A9      9E      0003A
03      01      50      78      0003E
      0000G      CF      54      E0      00042
                        00D3      31      00048

```

07	09	54	F0	0004B
	50	0000G	CF	9E 00050
09	00		50	F0 00055
50	6A	0003F460	8F	C9 0005A
	60		55	B0 00062

```

his builds
; a bit map of useable map entries. In the test which follows,
the maps
; to be used are checked for validity against this bit map.
;
CALLS #0, UNIBUS_SIZER ; 1030
;
; Invalidate all map entries and initialize the UET data register.
;
BISL3 #259170, UNIBUS_SPACE, R0 ; 1039
MNEGW #1, (R0) ; 1041
INSV #0, #31, #1, MAP_ENTRY ; 1042
CLRL N ; 1046
1$: MOVAL @UBI_UNDER_TEST[N], R1 ;
MOVL MAP_ENTRY, 2048(R1) ;
AOBLEQ #511, N, 1$ ; 1043
;
; Write a unique data pattern to a memory location, set up and
execute a
; DATI from that location, and check that the UET data register
contains the correct data. If it does not, then the correct
map has
; not been addressed properly due to a stuck or shorted Unibus
page frame
; bit. This sequence is repeated for each map entry of the set
of map
; entries 0, 1, 2, 4, 8, 16, 32, 64, 128, and 256.
;
CLRL M ; 1064
;
; Select the map number from the set 0, 1, 2, 4, 8, 16, 32, 64,
128,
; and 256. Check if the map selected is valid by checking the
valid
; map bitvector. If the map is useable, go ahead with the test
.
;
2$: MOVAB -1(R9), R0 ; 1075
ASHL R0, #1, N ;
BBS N, VALID_MAPS, 3$ ; 1076
BRW 6$ ;
;
; Set up the UET address register with the Unibus
address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.
;
3$: INSV N, #9, #7, UBUS_ADDR ; 1092
MOVAB SCRATCH, R0 ; 1093
INSV R0, #0, #9, UBUS_ADDR ;
BISL3 #259168, UNIBUS_SPACE, R0 ;
MOVW UBUS_ADDR, (R0) ; 1094
;
; Write the two MSBs of the Unibus address, which is the

```

```

; two MSBs of the map number.
;
58          54          02          07 EF 00065      EXTZV    #7, #2, N, TEMP3      ; 1105
          58          08 C4 0006A      MULL2    #8, TEMP3      ;
;
; Set up the UBI map. Put the CMI page frame number
; (PFN) into the register BUF_PFN. Then call the map
; setup routine with the parameters map number, PFN of
; the CMI buffer, byte offset mode (0 = no address
; offset), and data path number (0 is DDP). Also, set up
; the CMI source location.
;
52          56          56          0000G CF 9E 0006D      MOVAB    SCRATCH, TEMP1      ; 1120
          0F          09 EF 00072      EXTZV    #9, #15, TEMP1, BUF_PFN      ; 1121
          7E 7C 00077      CLRQ    -(SP)      ; 1122
          52 DD 00079      PUSHL   BUF_PFN      ;
          54 DD 0007B      PUSHL   N      ;
          0000G CF 04 FB 0007D      CALLS    #4, MAP_SETUP      ;
          0000G CF 59 D0 00082      MOVL     M, SCRATCH      ; 1123
;
; Execute the DATI, wait for a nominal period of time
; (100 us), and then read the UET data register.
; Compare the contents of the UET data register
; (which we have stored in register TEMP2) with the
; contents of the low word of the CMI location SCRATCH.
; If they are not the same, report the error.
;
50          6A 0003F464 8F C9 00087      BISL3    #259172, UNIBUS_SPACE, R0      ; 1136
60          58          01 A9 0008F      BISW3    #1, TEMP3, (R0)      ; 1138
          7E          0A 7D 00093      MOVQ     #10, -(SP)      ; 1140
          00000000G 9F          02 FB 00096      CALLS    #2, a#DS$WAITUS      ;
50          6A 0003F464 8F C9 0009D      BISL3    #259172, UNIBUS_SPACE, R0      ; 1141
          0000' CF 60 B0 000A5      MOVW     (R0), STATUS0      ;
          0000' CF FF1E 8F AA 000AA      BICW2    #65310, STATUS0      ;
          50          0000' CF 32 000B1      CVTWL   STATUS0, R0      ;
          1D 13 000B6      BEQL     4$      ;
          7E 7C 000B8      CLRQ    -(SP)      ;
          50 DD 000BA      PUSHL   R0      ;
          7E          02 7D 000BC      MOVQ     #2, -(SP)      ;
          0000G CF 9F 000BF      PUSHAB   FMT_UET_STAT      ;
          0000G CF 9F 000C3      PUSHAB   PRINT_GENERAL      ;
          0000G CF 9F 000C7      PUSHAB   UBIMOD_UB_CMI      ;
          7E          0000G CF 9A 000CB      MOVZBL   LUN, -(SP)      ;
          01 DD 000D0      PUSHL   #1      ;
          6B          0A FB 000D2      CALLS    #10, DS$ERRHARD      ;
          50          6A 0003F462 8F C9 000D5 4$: BISL3    #259170, UNIBUS_SPACE, R0      ; 1142
          57          60 B0 000DD      MOVW     (R0), TEMP2      ;
          10          00 ED 000E0      CMPZV    #0, #16, TEMP2, M      ; 1143
          1F 13 000E5      BEQL     5$      ;
          7E 7C 000E7      CLRQ    -(SP)      ; 1149
          57 3C 000E9      MOVZWL   TEMP2, -(SP)      ;
          59 DD 000EC      PUSHL   M      ;
          02 DD 000EE      PUSHL   #2      ;
          0000G CF 9F 000F0      PUSHAB   FMT_UB_CMI      ;
          0000G CF 9F 000F4      PUSHAB   PRINT_GENERAL      ;

```


ZZ-ECCBA-1.4
UBI_TESTS_6.10
01-04

TEST_008: Unibus to CMI Addressing Test
TEST_008: Unibus to CMI Addressing Test

D 2

6-Sep-1985

6-Sep-1985 17:19:10

6-Sep-1985 17:14:53

Fiche 2 Frame D2

VAX-11 Bliss-32 V4.1-746

[LEMIEUX.ECCBA.RELEASE]ECCBA4.B32;6

Sequence 222

Page 39
(5)

			0000G	CF	9F	000F8	PUSHAB	UBIMOD_UB_CMI	:
	7E		0000G	CF	9A	000FC	MOVZBL	LUN, -(SP)	:
				02	DD	00101	PUSHL	#2	:
	6B			0A	FB	00103	CALLS	#10, DS\$ERRHARD	:
	50		0000G	DF44	DE	00106	MOVAL	aUBI_UNDER_TEST[N], R0	: 1151
	0800			53	D0	0010C	MOVL	MAP_ENTRY, 2048(R0)	:
	00000000G			00	FB	00111	CALLS	#0, a#DS\$INLOOP	: 1156
				50	E9	00118	BLBC	R0, 6\$	:
				FF2D	31	0011B	BRW	3\$	:
FF16				09	F1	0011E	ACBL	#9, #1, M, 2\$	: 1064
	59			00	FB	00124	CALLS	#0, a#DS\$BREAK	: 1164
	00000000G			01	D0	0012B	MOVL	#1, R0	: 1166
				50		04	0012E	RET	:

; Routine Size: 303 bytes, Routine Base: TEST_008 + 0000


```
; 1167 2 $DS_BGNTEST (SECTION=(DEFAULT,ALL),TEXT='Data Path Select Test');
; 1168 2
; 1169 2 !++
; 1170 2 ! TEST DESCRIPTION:
; 1171 2 !
; 1172 2 ! This test detects any stuck or shorted bits in the data path select
; 1173 2 ! lines. The PURGE bit is written in each of the three BDP CSR's to
; 1174 2 ! ensure that the buffers are empty. Three DATO cycles are then
; 1175 2 ! initiated, with three unique data patterns, selecting each of the
; 1176 2 ! three different BDP's. With the three buffers loaded, another DATO
; 1177 2 ! transaction is initiated with a fourth unique pattern, selecting the
; 1178 2 ! DDP. Each BDP is purged in turn, and each written memory location
; 1179 2 ! is verified to contain the proper data pattern. Any stuck or shorted
; 1180 2 ! bit in the data path select lines will be clearly shown by one of the
; 1181 2 ! data patterns occurring in more than one location. The data patterns
; 1182 2 ! used are:
; 1183 2 !
; 1184 2 !           pattern 0      1122
; 1185 2 !           pattern 1      3344
; 1186 2 !           pattern 2      5566
; 1187 2 !           pattern 3      7788
; 1188 2 !
; 1189 2 ! Before the test is executed, the Unibus is sized for memory. Any
; 1190 2 ! Unibus memory present will cause map entries accessed with the same
; 1191 2 ! Unibus page frame to be unuseable. Therefore, before any testing
; 1192 2 ! requiring map functionality commences, the Unibus is sized, and a
; 1193 2 ! bit map of useable maps is built by the Unibus sizer routine. Any
; 1194 2 ! test requiring the use of one or more maps selects the maps to use on
; 1195 2 ! the basis of the entries in this bitvector.
; 1196 2 !
; 1197 2 ! ASSUMPTIONS:
; 1198 2 !
; 1199 2 !       Test 1-Test 6
; 1200 2 ! --
; 1201 2
; 1202 2 REGISTER
; 1203 2     BUF_PFN,
; 1204 2     MAP_NUMBER,
; 1205 2     N,
; 1206 2     UBUS_ADDR: WORD,
; 1207 2     TEMP,
; 1208 2     TEMP3;
; 1209 2
; 1210 2 MACRO
; 1211 2     UBUS_PF = UBUS_ADDR<9,7>%,      ! Defines the Unibus page frame field
; 1212 2     UBUS_BO = UBUS_ADDR<0,9>%;      ! Defines the Unibus byte number field
; 1213 2
; 1214 2 MAP
; 1215 2     SCRATCH: VECTOR[4,WORD];
; 1216 2
; 1217 2 CODECOMMENT
; 1218 2 '
; 1219 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 1220 2 'a bit map of useable map entries. In the test which follows, the maps',
; 1221 2 'selected are checked against this map for availability.'
```

ZZ-ECCBA-1.4
UBI_TESTS_6_10
01-04

TEST_009: Data Path Select Test
TEST_009: Data Path Select Test

F 2

6-Sep-1985

Fiche 2

Frame F2

Sequence 224

6-Sep-1985 17:19:10

VAX-11 Bliss-32 V4.1-746

Page 41

6-Sep-1985 17:14:53

[LEMIEUX.ECCBA.RELEASE]ECCBA4.B32;6

(6)

```
: 1222 2  ``:
: 1223 2
: 1224 3 BEGIN
: 1225 3
: 1226 3 UNIBUS_SIZER();
: 1227 3
: 1228 2 END;
: 1229 2
: 1230 2 CODECOMMENT
: 1231 2 ``:
: 1232 2 'Set up all of the data patterns in the table WORD_PATTERN.',
: 1233 2 ``:
: 1234 2
: 1235 3 BEGIN
: 1236 3
: 1237 3 WORD_PATTERN[0] = %X'1122';
: 1238 3 WORD_PATTERN[1] = %X'3344';
: 1239 3 WORD_PATTERN[2] = %X'5566';
: 1240 3 WORD_PATTERN[3] = %X'7788';
: 1241 3
: 1242 2 END;
: 1243 2
: 1244 2 CODECOMMENT
: 1245 2 ``:
: 1246 2 'Purge all of the buffered data paths, and clear the memory destinations.',
: 1247 2 ``:
: 1248 2
: 1249 3 BEGIN
: 1250 3
: 1251 3 INCR N FROM 1 TO 3 DO
: 1252 4 BEGIN
: 1253 4
: 1254 4 UBI_CSR(.N) = PURGE;
: 1255 4
: 1256 3 END;
: 1257 3
: 1258 3 SCRATCH[0] = 0;
: 1259 3 SCRATCH[1] = 0;
: 1260 3 SCRATCH[2] = 0;
: 1261 3 SCRATCH[3] = 0;
: 1262 3
: 1263 2 END;
: 1264 2
: 1265 2 CODECOMMENT
: 1266 2 ``:
: 1267 2 'Load UET data register with the pattern, and set up the',
: 1268 2 'UET address register and a UBI map. Then initiate a DAT0 and',
: 1269 2 'repeat for each buffered data path.',
: 1270 2 ``:
: 1271 2
: 1272 3 BEGIN
: 1273 3
: 1274 3 INCR N FROM 1 TO 3 DO
: 1275 4 BEGIN
: 1276 4
```

```
; 1277 4      CODECOMMENT
; 1278 4      ''
; 1279 4      'Set up the UET address register with the Unibus address.',
; 1280 4      'Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs of the',
; 1281 4      'UBI map number. Bits 0 to 8 contain the byte number field of',
; 1282 4      'the CMI address.',
; 1283 4      '';
; 1284 5      BEGIN
; 1285 5
; 1286 5      MAP_NUMBER = .FREE_MAP[.N];
; 1287 5      UET_DATA_REG = .WORD_PATTERN[.N];
; 1288 5      UBUS_PF = .MAP_NUMBER;
; 1289 5      UBUS_BO = SCRATCH[.N];
; 1290 5      UET_ADDR_REG = .UBUS_ADDR;
; 1291 5
; 1292 4      END;
; 1293 4
; 1294 4      CODECOMMENT
; 1295 4      ''
; 1296 4      'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
; 1297 4      'UBI map number.',
; 1298 4      '';
; 1299 5      BEGIN
; 1300 5
; 1301 5      TEMP3 = .MAP_NUMBER<7,2> * 8;
; 1302 5
; 1303 4      END;
; 1304 4
; 1305 4      CODECOMMENT
; 1306 4      ''
; 1307 4      'Set up the UBI map. Put the CMI page frame number (PFN) into',
; 1308 4      'the register BUF_PFN. Then call the map setup routine with the',
; 1309 4      'parameters map number, PFN of the CMI buffer, byte offset mode',
; 1310 4      '(0 = no byte offset), and data path number.',
; 1311 4      '';
; 1312 5      BEGIN
; 1313 5
; 1314 5      TEMP = SCRATCH[.N];
; 1315 5      BUF_PFN = .TEMP<9,15>;
; 1316 5      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.N);
; 1317 5
; 1318 5      UET_CTRL_REG = .TEMP3 OR DATO;
; 1319 5
; L 1320 5      UBI_STATUS(.N,UBIMOD_DP_SEL);
; xPRINT: Test 9, Subtest 0, Error 1
; L 1321 6      UET_STATUS(UBIMOD_DP_SEL);
; xPRINT: Test 9, Subtest 0, Error 2
; 1322 5
; 1323 4      END;
; 1324 4
; 1325 4      CODECOMMENT
; 1326 4      ''
; 1327 4      'Now make sure that the DATO did not write into memory -- it should',
; 1328 4      'only have written into the buffered data path.',
; 1329 4      '';
```

! M7

! M7


```

; 1330 5      BEGIN
; 1331 5
; 1332 5      IF .SCRATCH[.N] NEQ 0
; 1333 5      THEN
; 1334 6          BEGIN
; P 1335 6          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
; P 1336 6          PRLINK = PRINT_GENERAL,
; P 1337 6          P1 = FMT_DP_SEL,P2 = 2,
; L 1338 6          P3 = 0,P4 = .SCRATCH[.N]);
; xPRINT:      Test 9, Subtest 0, Error 3
; 1339 5          END;
; 1340 5
; 1341 4          END;
; 1342 4
; 1343 3      END;
; 1344 3
; 1345 2      END;
; 1346 2
; 1347 2      CODECOMMENT
; 1348 2      'Now each of the BDPs contain one word, and each BAR contains the',
; 1349 2      'indirect address of an element of SCRATCH. At this time, we set up',
; 1350 2      'for a DATO with the direct data path (DDP), and then execute a DATO.',
; 1351 2      '':
; 1352 2
; 1353 2      BEGIN
; 1354 3      CODECOMMENT
; 1355 3      'Set up the UET address register with the Unibus address.',
; 1356 3      'Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs of the',
; 1357 3      'UBI map number. Bits 0 to 8 contain the byte number field of',
; 1358 3      'the CMI address.',
; 1359 3      '':
; 1360 3      BEGIN
; 1361 4      MAP_NUMBER = .FREE_MAP[0];
; 1362 4      UBUS_PF = .MAP_NUMBER;
; 1363 4      UBUS_BO = SCRATCH[0];
; 1364 4      UET_ADDR_REG = .UBUS_ADDR;
; 1365 4
; 1366 4      END;
; 1367 4
; 1368 3      CODECOMMENT
; 1369 3      'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
; 1370 3      'UBI map number.',
; 1371 3      '':
; 1372 3      BEGIN
; 1373 4      TEMP3 = .MAP_NUMBER<7,2> * 8;
; 1374 4
; 1375 4      END;
; 1376 3      CODECOMMENT
; 1377 3
; 1378 3
; 1379 3
; 1380 3
; 1381 3
; 1382 3
; 1383 3

```

```

: 1384 3      ''
: 1385 3      'Set up the UBI map. Put the CMI page frame number (PFN) into',
: 1386 3      'the register BUF_PFN. Then call the map setup routine with the',
: 1387 3      'parameters map number, PFN of the CMI buffer, byte offset mode',
: 1388 3      '(0 = no address offset), and data path number.',
: 1389 3      ''
: 1390 4      BEGIN
: 1391 4
: 1392 4      TEMP = SCRATCH[0];
: 1393 4      BUF_PFN = .TEMP<9,15>;
: 1394 4      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,0);
: 1395 4
: 1396 3      END;
: 1397 3
: 1398 3      CODECOMMENT
: 1399 3      ''
: 1400 3      'Load the UET data register with the first pattern and execute',
: 1401 3      'the DATO. After a nominal wait, determine if the DATO worked. Also,',
: 1402 3      'verify that the other memory destinations are still empty.',
: 1403 3      ''
: 1404 4      BEGIN
: 1405 4
: 1406 4      UET_DATA_REG = .WORD_PATTERN[0];
: 1407 4
: 1408 4      UET_CTRL_REG = .TEMP3 OR DATO;
: 1409 4
: 1410 4      $DS_WAITUS(TIME=10);
: L 1411 5      UET_STATUS(UBIMOD_DP_SEL);          ! M7
: %PRINT:      Test 9, Subtest 0, Error 4
: 1412 4      IF .SCRATCH[0] NEQ .WORD_PATTERN[0]
: 1413 4      THEN
: 1414 5      BEGIN
: P 1415 5          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
: P 1416 5          PRLINK = PRINT_GENERAL,
: P 1417 5          P1 = FMT_DDP_DATO,P2 = 3,
: P 1418 5          P3 = SCRATCH[0],P4 = .WORD_PATTERN[0],
: L 1419 5          P5 = .SCRATCH[0]);! M10
: %PRINT:      Test 9, Subtest 0, Error 5
: 1420 4      END;
: 1421 4      INCR N FROM 1 TO 3 DO
: 1422 5      BEGIN
: 1423 5      IF .SCRATCH[.N] NEQ 0
: 1424 5      THEN
: 1425 6      BEGIN
: P 1426 6          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
: P 1427 6          PRLINK = PRINT_GENERAL,
: P 1428 6          P1 = FMT_DP_SEL,P2 = 2,
: L 1429 6          P3 = 0,P4 = .SCRATCH[.N]);
: %PRINT:      Test 9, Subtest 0, Error 6
: 1430 5      END;
: 1431 4      END;
: 1432 4
: 1433 3      END;
: 1434 3
: 1435 2      END;

```

```

: 1436 2
: 1437 2 CODECOMMENT
: 1438 2 ''
: 1439 2 'Now purge BDP1. This should write pattern 1 into SCRATCH element 1 only.',
: 1440 2 'Memory should then look like this:',
: 1441 2 ''
: 1442 2 '      SCRATCH[0]:      1122',
: 1443 2 '      SCRATCH[1]:      3344',
: 1444 2 '      SCRATCH[2]:      0000',
: 1445 2 '      SCRATCH[3]:      0000',
: 1446 2 ''
: 1447 3 BEGIN
: 1448 3
: 1449 3 UBI_CSR(1) = PURGE;
: 1450 3 $DS_WAITUS(TIME=10);
: 1451 3 IF .SCRATCH[0] NEQ .WORD_PATTERN[0]
: 1452 3 THEN
: 1453 4 BEGIN
: 1454 4 $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
: 1455 4 PRLINK = PRINT_GENERAL,
: 1456 4 P1 = FMT_DP_SEL,P2 = 2,
: 1457 4 P3 = .WORD_PATTERN[0],P4 = .SCRATCH[0]);
: 1458 3 *PRINT: Test 9, Subtest 0, Error 7
: 1459 3 END;
: 1460 3 IF .SCRATCH[1] NEQ .WORD_PATTERN[1]
: 1461 3 THEN
: 1462 4 BEGIN
: 1463 4 $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
: 1464 4 PRLINK = PRINT_GENERAL,
: 1465 4 P1 = FMT_DP_SEL,P2 = 2,
: 1466 4 P3 = .WORD_PATTERN[1],P4 = .SCRATCH[1]);
: 1467 3 *PRINT: Test 9, Subtest 0, Error 8
: 1468 3 END;
: 1469 3 INCR N FROM 2 TO 3 DO
: 1470 4 BEGIN
: 1471 4 IF .SCRATCH[.N] NEQ 0
: 1472 4 THEN
: 1473 5 BEGIN
: 1474 5 $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
: 1475 5 PRLINK = PRINT_GENERAL,
: 1476 5 P1 = FMT_DP_SEL,P2 = 2,
: 1477 5 P3 = 0,P4 = .SCRATCH[.N]);
: 1478 4 *PRINT: Test 9, Subtest 0, Error 9
: 1479 4 END;
: 1480 3 END;
: 1481 2 END;
: 1482 2 CODECOMMENT
: 1483 2 ''
: 1484 2 'Now purge BDP2. This should write pattern 2 into SCRATCH element 2 only.',
: 1485 2 'Memory should then look like this:',
: 1486 2 ''
: 1487 2 '      SCRATCH[0]:      1122',

```



```

1488 2      :      SCRATCH[1]:      3344',
1489 2      :      SCRATCH[2]:      5566',
1490 2      :      SCRATCH[3]:      0000',
1491 2      :
1492 3      :      BEGIN
1493 3
1494 3      :      UBI_CSR(2) = PURGE;
1495 3      :      $DS_WAITUS(TIME=10);
1496 3      :      IF .SCRATCH[0] NEQ .WORD_PATTERN[0]
1497 3      :      THEN
1498 4          BEGIN
1499 4          :      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
1500 4          :      PRLINK = PRINT_GENERAL,
1501 4          :      P1 = FMT_DP_SEL,P2 = 2,
1502 4          :      P3 = .WORD_PATTERN[0],P4 = .SCRATCH[0]);
1503 3      :      xPRINT:
1504 3      :      Test 9, Subtest 0, Error 10
1505 3      :      END;
1506 4          BEGIN
1507 4          :      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
1508 4          :      PRLINK = PRINT_GENERAL,
1509 4          :      P1 = FMT_DP_SEL,P2 = 2,
1510 4          :      P3 = .WORD_PATTERN[1],P4 = .SCRATCH[1]);
1511 3      :      xPRINT:
1512 3      :      Test 9, Subtest 0, Error 11
1513 3      :      END;
1514 4          BEGIN
1515 4          :      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
1516 4          :      PRLINK = PRINT_GENERAL,
1517 4          :      P1 = FMT_DP_SEL,P2 = 2,
1518 4          :      P3 = .WORD_PATTERN[2],P4 = .SCRATCH[2]);
1519 3      :      xPRINT:
1520 3      :      Test 9, Subtest 0, Error 12
1521 3      :      END;
1522 4          BEGIN
1523 4          :      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
1524 4          :      PRLINK = PRINT_GENERAL,
1525 4          :      P1 = FMT_DP_SEL,P2 = 2,
1526 4          :      P3 = 0,P4 = .SCRATCH[3]);
1527 3      :      xPRINT:
1528 3      :      Test 9, Subtest 0, Error 13
1529 2      :      END;
1530 2
1531 2      :      CODECOMMENT
1532 2      :      'Now purge BDP3. This should write pattern 3 into SCRATCH element 3 only.',
1533 2      :      'Memory should then look like this:',
1534 2      :      '
1535 2      :      SCRATCH[0]:      1122',
1536 2      :      SCRATCH[1]:      3344',
1537 2      :      SCRATCH[2]:      5566',
1538 2

```

```

: 1539 2      ' SCRATCH[3]: 7788',
: 1540 2      '::
: 1541 3      BEGIN
: 1542 3
: 1543 3      UBI_CSR(3) = PURGE;
: 1544 3      $DS_WAITUS(TIME=10);
: 1545 3      IF SCRATCH[0] NEQ WORD_PATTERN[0]
: 1546 3      THEN
: 1547 4          BEGIN
: P 1548 4              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
: P 1549 4                  PRLINK = PRINT_GENERAL,
: P 1550 4                  P1 = FMT_DP_SEL,P2 = 2,
: L 1551 4                  P3 = WORD_PATTERN[0],P4 = SCRATCH[0]);
: xPRINT:      Test 9, Subtest 0, Error 14
: 1552 3      END;
: 1553 3      IF SCRATCH[1] NEQ WORD_PATTERN[1]
: 1554 3      THEN
: 1555 4          BEGIN
: P 1556 4              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
: P 1557 4                  PRLINK = PRINT_GENERAL,
: P 1558 4                  P1 = FMT_DP_SEL,P2 = 2,
: L 1559 4                  P3 = WORD_PATTERN[1],P4 = SCRATCH[1]);
: xPRINT:      Test 9, Subtest 0, Error 15
: 1560 3      END;
: 1561 3      IF SCRATCH[2] NEQ WORD_PATTERN[2]
: 1562 3      THEN
: 1563 4          BEGIN
: P 1564 4              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
: P 1565 4                  PRLINK = PRINT_GENERAL,
: P 1566 4                  P1 = FMT_DP_SEL,P2 = 2,
: L 1567 4                  P3 = WORD_PATTERN[2],P4 = SCRATCH[2]);
: xPRINT:      Test 9, Subtest 0, Error 16
: 1568 3      END;
: 1569 3      IF SCRATCH[3] NEQ WORD_PATTERN[3]
: 1570 3      THEN
: 1571 4          BEGIN
: P 1572 4              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DP_SEL,
: P 1573 4                  PRLINK = PRINT_GENERAL,
: P 1574 4                  P1 = FMT_DP_SEL,P2 = 2,
: L 1575 4                  P3 = WORD_PATTERN[3],P4 = SCRATCH[3]);
: xPRINT:      Test 9, Subtest 0, Error 17
: 1576 3      END;
: 1577 3
: 1578 2      END;
: 1579 2
: 1580 1      $DS_ENDTEST;

```

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00048 \$:
00000000 00000003 00058

.ADDRESS P.AAJ, P.AAK, P.AAL, TEST_009
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

ZZ-ECCBA-1.4
UBI_TESTS_6_10
01-04

TEST_009: Data Path Select Test
TEST_009: Data Path Select Test

M 2

6-Sep-1985

Fiche 2

Frame M2

Sequence 231

6-Sep-1985 17:19:10

VAX-11 Bliss-32 V4.1-746

Page 48

6-Sep-1985 17:14:53

[LEMIEUX.ECCBA.RELEASE]ECCBA4.B32;6

(6)

```
65 6C 65 53 20 68 74 61 50 20 61 74 61 44 15 0009 00064 P.AAJ: .WORD 9 ;
74 73 65 54 20 74 63 00066 P.AAK: .ASCII <21>\Data Path Select Test\ ;
00000000 00075 ;
0007C P.AAL: .LONG 0 ;

.PSECT TEST_009,NOWRT,2
.OFFC 00000
.ENTRY TEST_009, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1167
R11
MOVAB UBIMOD_DP_SEL, R11
MOVAB SCRATCH, R10
MOVAB WORD_PATTERN, R9
MOVAB a#DS$ERRHARD, R8
; Call the Unibus sizer routine to locate any Unibus memory. This builds
; a bit map of useable map entries. In the test which follows,
; the maps
; selected are checked against this map for availability.
0000G CF 00 FB 00018 CALLS #0, UNIBUS_SIZER ; 1226
; Set up all of the data patterns in the table WORD_PATTERN.
04 69 33441122 8F D0 0001D MOVL #860098850, WORD_PATTERN ; 1237
A9 77885566 8F D0 00024 MOVL #2005423462, WORD_PATTERN+4 ; 1239
; Purge all of the buffered data paths, and clear the memory destinations.
F6 50 0000GDF40 01 D0 0002C 1$: MOVL #1, N ; 1254
50 01 D0 0002F MOVL #1, aUBI_UNDER_TEST[N] ;
03 F3 00035 AOBLEQ #3, N, 1$ ; 1251
6A 7C 00039 CLRQ SCRATCH ; 1258
; Load UET data register with the pattern, and set up the
; UET address register and a UBI map. Then initiate a DATO and
; repeat for each buffered data path.
57 01 D0 0003B MOVL #1, N ; 1274
; Set up the UET address register with the Unibus address.
; Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs of the
; UBI map number. Bits 0 to 8 contain the byte number field of
; the CMI address.
50 53 0000GCF47 3C 0003E 2$: MOVZWL FREE_MAP[N], MAP_NUMBER ; 1286
0000G CF 0003F462 8F C9 00044 BISL3 #259170, UNIBUS_SPACE, R0 ;
60 6947 B0 0004E MOVW WORD_PATTERN[N], (R0) ; 1287
09 53 F0 00052 INSV MAP_NUMBER, #9, #7, UBUS_ADDR ; 1288
50 6A47 3E 00057 MOVAV SCRATCH[N], R0 ; 1289
```



```

54      09      00      50  F0 0005B      INSV      R0, #0, #9, UBUS_ADDR      ;
      50      0000G  CF 0003F460 8F  C9 00060      BISL3      #259168, UNIBUS_SPACE, R0      ;
      60      60      54  B0 0006A      MOVW      UBUS_ADDR, (R0)      ; 1290
;
; Write the 2 MSBs of the Unibus address with the 2 MSBs of the
; UBI map number.
;
56      53      02      07  EF 0006D      EXTZV      #7, #2, MAP_NUMBER, TEMP3      ; 1301
      56      56      08  C4 00072      MULL2      #8, TEMP3      ;
;
; Set up the UBI map. Put the CMI page frame number (PFN) into
; the register BUF_PFN. Then call the map setup routine with t
; he
; parameters map number, PFN of the CMI buffer, byte offset mod
; e
; (0 = no byte offset), and data path number.
;
52      55      55      6A47 3E 00075      MOVAV      SCRATCH[N], TEMP      ; 1314
      55      0F      09  EF 00079      EXTZV      #9, #15, TEMP, BUF_PFN      ; 1315
      57      DD 0007E      PUSHL      N      ; 1316
      7E      D4 00080      CLRL      -(SP)      ;
      52      DD 00082      PUSHL      BUF_PFN      ;
      53      DD 00084      PUSHL      MAP_NUMBER      ;
      04      FB 00086      CALLS      #4, MAP_SETUP      ;
      50      0000G  CF 0003F464 8F  C9 0008B      BISL3      #259172, UNIBUS_SPACE, R0      ;
      60      56      05  A9 00095      BISW3      #5, TEMP3, (R0)      ; 1318
      FC  A9      0000GDF47 1FFFFFFE 8F  CB 00099      BICL3      #536870910, aUBI_UNDER_TEST[N], STATUS1      ; 1320
      50      50      A9  D0 000A5      MOVL      STATUS1, R0      ;
      1A      18 000A9      BGEQ      3$      ;
      7E      7C 000AB      CLRQ      -(SP)      ;
      7E      D4 000AD      CLRL      -(SP)      ;
      50      DD 000AF      PUSHL      R0      ;
      7E      D4 000B1      CLRL      -(SP)      ;
      57      DD 000B3      PUSHL      N      ;
      0000G  CF 9F 000B5      PUSHAB     PRINT_CSR_ERR      ;
      5B      DD 000B9      PUSHL      R11      ;
      7E      0000G  CF 9A 000BB      MOVZBL     LUN, -(SP)      ;
      01      DD 000C0      PUSHL      #1      ;
      68      0A  FB 000C2      CALLS      #10, DS$ERRHARD      ;
      50      0000G  CF 0003F464 8F  C9 000C5 3$:  BISL3      #259172, UNIBUS_SPACE, R0      ; 1321
      F8      A9      60  B0 000CF      MOVW      (R0), STATUS0      ;
      F8      A9      8F  AA 000D3      BICW2      #65310, STATUS0      ;
      50      FF1E 8F  A9 32 000D9      CVTWL      STATUS0, R0      ;
      F8      A9      1B  13 000DD      BEQL      4$      ;
      7E      7C 000DF      CLRQ      -(SP)      ;
      50      DD 000E1      PUSHL      R0      ;
      7E      02  7D 000E3      MOVQ      #2, -(SP)      ;
      0000G  CF 9F 000E6      PUSHAB     FMT_UET_STAT      ;
      0000G  CF 9F 000EA      PUSHAB     PRINT_GENERAL      ;
      5B      DD 000EE      PUSHL      R11      ;
      7E      0000G  CF 9A 000F0      MOVZBL     LUN, -(SP)      ;
      02      DD 000F5      PUSHL      #2      ;
      68      0A  FB 000F7      CALLS      #10, DS$ERRHARD      ;

```

; Now make sure that the DAT0 did not write into memory -- it s

;

ould
; only have written into the buffered data path.

		50		6A47	3C	000FA	4\$:	MOVZWL	SCRATCH[N], R0		; 1332
				1B	13	000FE		BEQL	5\$		; 1338
				7E	7C	00100		CLRQ	-(SP)		
				50	DD	00102		PUSHL	R0		
		7E		02	7D	00104		MOVQ	#2, -(SP)		
			0000G	CF	9F	00107		PUSHAB	FMT_DP_SEL		
			0000G	CF	9F	0010B		PUSHAB	PRINT_GENERAL		
				5B	DD	0010F		PUSHL	R11		
		7E	0000G	CF	9A	00111		MOVZBL	LUN, -(SP)		
				03	DD	00116		PUSHL	#3		
		68		0A	FB	00118		CALLS	#10, DS\$ERRHARD		
FF1D	57	01		03	F1	0011B	5\$:	ACBL	#3, #1, N, 2\$		; 1274

; Now each of the BDPs contain one word, and each BAR contains
the
; indirect address of an element of SCRATCH. At this time, we
set up
; for a DATO with the direct data path (DDP), and then execute
a DATO.

; Set up the UET address register with the Unibus address.
; Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs of th
e
; UBI map number. Bits 0 to 8 contain the byte number field of
the CMI address.

		53	0000G	CF	3C	00121		MOVZWL	FREE_MAP, MAP_NUMBER		; 1365
54	07	09		53	F0	00126		INSV	MAP_NUMBER, #9, #7, UBUS_ADDR		; 1366
		50		6A	9E	0012B		MOVAB	SCRATCH, R0		; 1367
54	09	00		50	F0	0012E		INSV	R0, #0, #9, UBUS_ADDR		
	50	0000G	CF	0003F460	8F	C9	00133	BISL3	#259168, UNIBUS_SPACE, R0		
		60		54	B0	0013D		MOVW	UBUS_ADDR, (R0)		; 1368

; Write the 2 MSBs of the Unibus address with the 2 MSBs of the
UBI map number.

		56		02	07	EF	00140	EXTZV	#7, #2, MAP_NUMBER, TEMP3		; 1379
		56		08	C4	00145		MULL2	#8, TEMP3		

; Set up the UBI map. Put the CMI page frame number (PFN) into
the register BUF_PFN. Then call the map setup routine with t
he
; parameters map number, PFN of the CMI buffer, byte offset mod
e
; (0 = no address offset), and data path number.

		55		6A	9E	00148		MOVAB	SCRATCH, TEMP		; 1392
52	55	0F		09	EF	0014B		EXTZV	#9, #15, TEMP, BUF_PFN		; 1393
				7E	7C	00150		CLRQ	-(SP)		; 1394
				52	DD	00152		PUSHL	BUF_PFN		
				53	DD	00154		PUSHL	MAP_NUMBER		


```

0000G CF      04 FB 00156      CALLS #4, MAP_SETUP      ;
;
; Load the UET data register with the first pattern and execute
; the DATO. After a nominal wait, determine if the DATO worked
; Also,
; verify that the other memory destinations are still empty.
;
50      0000G CF 0003F462 8F C9 0015B      BISL3 #259170, UNIBUS_SPACE, R0      ; 1404
60      0000G      69 B0 00165      MOVW WORD_PATTERN, (R0)      ; 1406
50      0000G CF 0003F464 8F C9 00168      BISL3 #259172, UNIBUS_SPACE, R0      ;
60      56      05 A9 00172      BSW3 #5, TEMP3, (R0)      ; 1408
7E      0A 7D 00176      MOVQ #10, -(SP)      ; 1410
00000000G 9F      02 FB 00179      CALLS #2, @DS$WAITUS      ;
50      0000G CF 0003F464 8F C9 00180      BISL3 #259172, UNIBUS_SPACE, R0      ; 1411
F8      A9      60 B0 0018A      MOVW (R0), STATUS0      ;
F8      A9      8F AA 0018E      BICW2 #65310, STATUS0      ;
50      FF1E 8F A9 32 00194      CVTWL STATUS0, R0      ;
F8      A9      1B 13 00198      BEQL 6$      ;
7E      7E 7C 0019A      CLRQ -(SP)      ;
50      DD 0019C      PUSHL R0      ;
7E      02 7D 0019E      MOVQ #2, -(SP)      ;
0000G CF 9F 001A1      PUSHAB FMT_UET_STAT      ;
0000G CF 9F 001A5      PUSHAB PRINT_GENERAL      ;
5B      DD 001A9      PUSHL R11      ;
7E      0000G CF 9A 001AB      MOVZBL LUN, -(SP)      ;
04      DD 001B0      PUSHL #4      ;
68      0A FB 001B2      CALLS #10, DS$ERRHARD      ;
69      6A B1 001B5 6$: CMPW SCRATCH, WORD_PATTERN      ; 1412
20      13 001B8      BEQL 7$      ;
7E      D4 001BA      CLRL -(SP)      ; 1419
7E      6A 3C 001BC      MOVZWL SCRATCH, -(SP)      ;
7E      69 3C 001BF      MOVZWL WORD_PATTERN, -(SP)      ;
5A      DD 001C2      PUSHL R10      ;
03      DD 001C4      PUSHL #3      ;
0000G CF 9F 001C6      PUSHAB FMT_DDP_DATO      ;
0000G CF 9F 001CA      PUSHAB PRINT_GENERAL      ;
5B      DD 001CE      PUSHL R11      ;
7E      0000G CF 9A 001D0      MOVZBL LUN, -(SP)      ;
05      DD 001D5      PUSHL #5      ;
68      0A FB 001D7      CALLS #10, DS$ERRHARD      ;
52      01 D0 001DA 7$: MOVL #1, N      ; 1421
50      6A42 3C 001DD 8$: MOVZWL SCRATCH[N], R0      ; 1423
1B      13 001E1      BEQL 9$      ;
7E      7C 001E3      CLRQ -(SP)      ; 1429
50      DD 001E5      PUSHL R0      ;
7E      02 7D 001E7      MOVQ #2, -(SP)      ;
0000G CF 9F 001EA      PUSHAB FMT_DP_SEL      ;
0000G CF 9F 001EE      PUSHAB PRINT_GENERAL      ;
5B      DD 001F2      PUSHL R11      ;
7E      0000G CF 9A 001F4      MOVZBL LUN, -(SP)      ;
06      DD 001F9      PUSHL #6      ;
68      0A FB 001FB      CALLS #10, DS$ERRHARD      ;
DB      52      03 F3 001FE 9$: AOBLEQ #3, N, 8$      ; 1421
;
; Now purge BDP1. This should write pattern 1 into SCRATCH ele

```


;

ment 1 only.
; Memory should then look like this:

```

;
;   SCRATCH[0]:    1122
;   SCRATCH[1]:    3344
;   SCRATCH[2]:    0000
;   SCRATCH[3]:    0000
;

```

	50	0000G	CF	D0	00202	MOVL	UBI_UNDER_TEST, R0		; 1449
04	A0		01	D0	00207	MOVL	#1, 4(R0)		; 1450
	7E		0A	7D	0020B	MOVQ	#10, -(SP)		; 1451
00000000G	9F		02	FB	0020E	CALLS	#2, @DS\$WAITUS		; 1457
	69		6A	B1	00215	CMPW	SCRATCH, WORD_PATTERN		; 1459
			1E	13	00218	BEQL	10\$		; 1465
			7E	7C	0021A	CLRQ	-(SP)		; 1466
	7E		6A	3C	0021C	MOVZWL	SCRATCH, -(SP)		; 1467
	7E		69	3C	0021F	MOVZWL	WORD_PATTERN, -(SP)		; 1468
			02	DD	00222	PUSHL	#2		; 1469
		0000G	CF	9F	00224	PUSHAB	FMT_DP_SEL		; 1470
		0000G	CF	9F	00228	PUSHAB	PRINT_GENERAL		; 1471
			5B	DD	0022C	PUSHL	R11		; 1472
	7E	0000G	CF	9A	0022E	MOVZBL	LUN, -(SP)		; 1473
			07	DD	00233	PUSHL	#7		; 1474
	68		0A	FB	00235	CALLS	#10, DS\$ERRHARD		; 1475
02	A9	02	AA	B1	00238	CMPW	SCRATCH+2, WORD_PATTERN+2		; 1476
			20	13	0023D	BEQL	11\$		; 1477
			7E	7C	0023F	CLRQ	-(SP)		; 1478
	7E	02	AA	3C	00241	MOVZWL	SCRATCH+2, -(SP)		; 1479
	7E	02	A9	3C	00245	MOVZWL	WORD_PATTERN+2, -(SP)		; 1480
			02	DD	00249	PUSHL	#2		; 1481
		0000G	CF	9F	0024B	PUSHAB	FMT_DP_SEL		; 1482
		0000G	CF	9F	0024F	PUSHAB	PRINT_GENERAL		; 1483
			5B	DD	00253	PUSHL	R11		; 1484
	7E	0000G	CF	9A	00255	MOVZBL	LUN, -(SP)		; 1485
			08	DD	0025A	PUSHL	#8		; 1486
	68		0A	FB	0025C	CALLS	#10, DS\$ERRHARD		; 1487
	52		02	D0	0025F	MOVL	#2, N		; 1488
	50		6A42	3C	00262	MOVZWL	SCRATCH[N], R0		; 1489
			1B	13	00266	BEQL	13\$		; 1490
			7E	7C	00268	CLRQ	-(SP)		; 1491
			50	DD	0026A	PUSHL	R0		; 1492
	7E		02	7D	0026C	MOVQ	#2, -(SP)		; 1493
		0000G	CF	9F	0026F	PUSHAB	FMT_DP_SEL		; 1494
		0000G	CF	9F	00273	PUSHAB	PRINT_GENERAL		; 1495
			5B	DD	00277	PUSHL	R11		; 1496
	7E	0000G	CF	9A	00279	MOVZBL	LUN, -(SP)		; 1497
			09	DD	0027E	PUSHL	#9		; 1498
	68		0A	FB	00280	CALLS	#10, DS\$ERRHARD		; 1499
DB	52		03	F3	00283	AOBLEQ	#3, N, 12\$		; 1500

; Now purge BDP2. This should write pattern 2 into SCRATCH element 2 only.

; Memory should then look like this:

```

;
;   SCRATCH[0]:    1122
;

```

;

```

;          SCRATCH[1]:      3344
;          SCRATCH[2]:      5566
;          SCRATCH[3]:      0000
;
08 50      0000G CF D0 00287      MOVL      UBI_UNDER_TEST, R0      ; 1494
    A0      01 D0 0028C      MOVL      #1, 8(R0)      ;
    7E      0A 7D 00290      MOVQ      #10, -(SP)      ; 1495
00000000G 9F      02 FB 00293      CALLS     #2, a#DS$WAITUS      ;
    69      6A B1 0029A      CMPW      SCRATCH, WORD_PATTERN      ; 1496
    1E      13 0029D      BEQL      14$      ;
    7E      7E 7C 0029F      CLRQ      -(SP)      ; 1502
    7E      6A 3C 002A1      MOVZWL     SCRATCH, -(SP)      ;
    7E      69 3C 002A4      MOVZWL     WORD_PATTERN, -(SP)      ;
    02      DD 002A7      PUSHL      #2      ;
    0000G    CF 9F 002A9      PUSHAB     FMT_DP_SEL      ;
    0000G    CF 9F 002AD      PUSHAB     PRINT_GENERAL      ;
    5B      DD 002B1      PUSHL      R11      ;
    7E      0000G CF 9A 002B3      MOVZBL     LUN, -(SP)      ;
    0A      DD 002B8      PUSHL      #10      ;
    68      0A FB 002BA      CALLS     #10, DS$ERRHARD      ;
02  A9      02 AA B1 002BD 14$:    CMPW      SCRATCH+2, WORD_PATTERN+2      ; 1504
    20      13 002C2      BEQL      15$      ;
    7E      7E 7C 002C4      CLRQ      -(SP)      ; 1510
    7E      02 AA 3C 002C6      MOVZWL     SCRATCH+2, -(SP)      ;
    7E      02 A9 3C 002CA      MOVZWL     WORD_PATTERN+2, -(SP)      ;
    02      DD 002CE      PUSHL      #2      ;
    0000G    CF 9F 002D0      PUSHAB     FMT_DP_SEL      ;
    0000G    CF 9F 002D4      PUSHAB     PRINT_GENERAL      ;
    5B      DD 002D8      PUSHL      R11      ;
    7E      0000G CF 9A 002DA      MOVZBL     LUN, -(SP)      ;
    0B      DD 002DF      PUSHL      #11      ;
    68      0A FB 002E1      CALLS     #10, DS$ERRHARD      ;
04  A9      04 AA B1 002E4 15$:    CMPW      SCRATCH+4, WORD_PATTERN+4      ; 1512
    20      13 002E9      BEQL      16$      ;
    7E      7E 7C 002EB      CLRQ      -(SP)      ; 1518
    7E      04 AA 3C 002ED      MOVZWL     SCRATCH+4, -(SP)      ;
    7E      04 A9 3C 002F1      MOVZWL     WORD_PATTERN+4, -(SP)      ;
    02      DD 002F5      PUSHL      #2      ;
    0000G    CF 9F 002F7      PUSHAB     FMT_DP_SEL      ;
    0000G    CF 9F 002FB      PUSHAB     PRINT_GENERAL      ;
    5B      DD 002FF      PUSHL      R11      ;
    7E      0000G CF 9A 00301      MOVZBL     LUN, -(SP)      ;
    0C      DD 00306      PUSHL      #12      ;
    68      0A FB 00308      CALLS     #10, DS$ERRHARD      ;
    50      06 AA 3C 0030B 16$:    MOVZWL     SCRATCH+6, R0      ; 1520
    1B      13 0030F      BEQL      17$      ;
    7E      7E 7C 00311      CLRQ      -(SP)      ; 1526
    50      DD 00313      PUSHL      R0      ;
    7E      02 7D 00315      MOVQ      #2, -(SP)      ;
    0000G    CF 9F 00318      PUSHAB     FMT_DP_SEL      ;
    0000G    CF 9F 0031C      PUSHAB     PRINT_GENERAL      ;
    5B      DD 00320      PUSHL      R11      ;
    7E      0000G CF 9A 00322      MOVZBL     LUN, -(SP)      ;
    0D      DD 00327      PUSHL      #13      ;
    68      0A FB 00329      CALLS     #10, DS$ERRHARD      ;

```

				; Now purge BDP3. This should write pattern 3 into SCRATCH element 3 only.			
				; Memory should then look like this:			
				; SCRATCH[0]: 1122			
				; SCRATCH[1]: 3344			
				; SCRATCH[2]: 5566			
				; SCRATCH[3]: 7788			
				; 17\$:			
0C	50	0000G	CF D0 0032C	MOVL	UBI_UNDER_TEST, R0	:	1543
	A0		01 D0 00331	MOVL	#1, 12(R0)	:	
	7E		0A 7D 00335	MOVQ	#10, -(SP)	:	1544
00000000G	9F		02 FB 00338	CALLS	#2, a#DS\$WAITUS	:	
	69		6A B1 0033F	CMPW	SCRATCH, WORD_PATTERN	:	1545
			1E 13 00342	BEQL	18\$	:	
			7E 7C 00344	CLRQ	-(SP)	:	1551
	7E		6A 3C 00346	MOVZWL	SCRATCH, -(SP)	:	
	7E		69 3C 00349	MOVZWL	WORD_PATTERN, -(SP)	:	
			02 DD 0034C	PUSHL	#2	:	
		0000G	CF 9F 0034E	PUSHAB	FMT_DP_SEL	:	
		0000G	CF 9F 00352	PUSHAB	PRINT_GENERAL	:	
			5B DD 00356	PUSHL	R11	:	
	7E	0000G	CF 9A 00358	MOVZBL	LUN, -(SP)	:	
			0E DD 0035D	PUSHL	#14	:	
02	68		0A FB 0035F	CALLS	#10, DS\$ERRHARD	:	
	A9	02	AA B1 00362	CMPW	SCRATCH+2, WORD_PATTERN+2	:	1553
			20 13 00367	BEQL	19\$	:	
			7E 7C 00369	CLRQ	-(SP)	:	1559
	7E	02	AA 3C 0036B	MOVZWL	SCRATCH+2, -(SP)	:	
	7E	02	A9 3C 0036F	MOVZWL	WORD_PATTERN+2, -(SP)	:	
			02 DD 00373	PUSHL	#2	:	
		0000G	CF 9F 00375	PUSHAB	FMT_DP_SEL	:	
		0000G	CF 9F 00379	PUSHAB	PRINT_GENERAL	:	
			5B DD 0037D	PUSHL	R11	:	
	7E	0000G	CF 9A 0037F	MOVZBL	LUN, -(SP)	:	
			0F DD 00384	PUSHL	#15	:	
04	68		0A FB 00386	CALLS	#10, DS\$ERRHARD	:	
	A9	04	AA B1 00389	CMPW	SCRATCH+4, WORD_PATTERN+4	:	1561
			20 13 0038E	BEQL	20\$	:	
			7E 7C 00390	CLRQ	-(SP)	:	1567
	7E	04	AA 3C 00392	MOVZWL	SCRATCH+4, -(SP)	:	
	7E	04	A9 3C 00396	MOVZWL	WORD_PATTERN+4, -(SP)	:	
			02 DD 0039A	PUSHL	#2	:	
		0000G	CF 9F 0039C	PUSHAB	FMT_DP_SEL	:	
		0000G	CF 9F 003A0	PUSHAB	PRINT_GENERAL	:	
			5B DD 003A4	PUSHL	R11	:	
	7E	0000G	CF 9A 003A6	MOVZBL	LUN, -(SP)	:	
			10 DD 003AB	PUSHL	#16	:	
06	68		0A FB 003AD	CALLS	#10, DS\$ERRHARD	:	
	A9	06	AA B1 003B0	CMPW	SCRATCH+6, WORD_PATTERN+6	:	1569
			20 13 003B5	BEQL	21\$	:	
			7E 7C 003B7	CLRQ	-(SP)	:	1575
	7E	06	AA 3C 003B9	MOVZWL	SCRATCH+6, -(SP)	:	
	7E	06	A9 3C 003BD	MOVZWL	WORD_PATTERN+6, -(SP)	:	

ZZ-ECCBA-1.4
UBI TESTS_6_10
01-04

TEST_009: Data Path Select Test

TEST_009: Data Path Select Test

G 3

6-Sep-1985

Fiche 2

Frame G3

Sequence 238

6-Sep-1985 17:19:10

VAX-11 Bliss-32 V4.1-746

Page 55

6-Sep-1985 17:14:53

[LEMIEUX.ECCBA.RELEASE]ECCBA4.B32;6

(6)

		02	DD	003C1	PUSHL	#2	:
	0000G	CF	9F	003C3	PUSHAB	FMT_DP_SEL	:
	0000G	CF	9F	003C7	PUSHAB	PRINT_GENERAL	:
		5B	DD	003CB	PUSHL	R11	:
	7E	0000G	CF	9A	MOVZBL	LUN, -(SP)	:
		11	DD	003D2	PUSHL	#17	:
	68	0A	FB	003D4	CALLS	#10, DS\$ERRHARD	:
00000000G	9F	00	FB	003D7	CALLS	#0, @DS\$BREAK	: 1578
	50	01	D0	003DE	MOVL	#1, R0	: 1580
		04	003E1	RET			:

; Routine Size: 994 bytes, Routine Base: TEST_009 + 0000

```
; 1581 2 $DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Direct Data Path DATI Test');
; 1582 2
; 1583 2 !++
; 1584 2 ! TEST DESCRIPTION:
; 1585 2 !
; 1586 2 ! This tests the ability of the UBI to perform a DATI transaction
; 1587 2 ! through the direct data path (DDP). The transactions are made from
; 1588 2 ! longword and word aligned CMI addresses. The following data patterns
; 1589 2 ! are used:
; 1590 2 !
; 1591 2 ! pattern 1 0101010101010101
; 1592 2 ! pattern 2 1010101010101010
; 1593 2 ! pattern 3 0011001100110011
; 1594 2 ! pattern 4 0000111100001111
; 1595 2 ! pattern 5 0000000011111111
; 1596 2 !
; 1597 2 ! Before testing starts, the Unibus Sizer routine is called to determine
; 1598 2 ! which maps are useable.
; 1599 2 !
; 1600 2 ! ASSUMPTIONS:
; 1601 2 !
; 1602 2 ! Test 1-Test 7
; 1603 2 ! --
; 1604 2
; 1605 2 REGISTER
; 1606 2 BUF_PFN,
; 1607 2 M,
; 1608 2 MAP_NUMBER,
; 1609 2 N,
; 1610 2 TEMP,
; 1611 2 TEMP1: WORD,
; 1612 2 TEMP3,
; 1613 2 UBUS_ADDR: WORD;
; 1614 2
; 1615 2 MACRO
; 1616 2 UBUS_PF = UBUS_ADDR<9,7>%, ! Defines the Unibus page frame field
; 1617 2 UBUS_BO = UBUS_ADDR<0,9>%; ! Defines the Unibus byte number field
; 1618 2
; 1619 2 MAP
; 1620 2 SCRATCH: VECTOR[5,WORD];
; 1621 2
; 1622 2 CODECOMMENT
; 1623 2 ''
; 1624 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 1625 2 'a table of useable map entries. In the test which follows, the map',
; 1626 2 'selected is drawn from this table.',
; 1627 2 '';
; 1628 3 BEGIN
; 1629 3
; 1630 3 UNIBUS_SIZER();
; 1631 3 MAP_NUMBER = .FREE_MAP[0];
; 1632 3
; 1633 2 END;
; 1634 2
; 1635 2 CODECOMMENT
```

```

; 1636 2  ''
; 1637 2  'Load the data patterns into the table WORD_PATTERN for later use.',
; 1638 2  ''
; 1639 3  BEGIN
; 1640 3
; 1641 3  WORD_PATTERN[0] = %X'5555';
; 1642 3  WORD_PATTERN[1] = %X'AAAA';
; 1643 3  WORD_PATTERN[2] = %X'3333';
; 1644 3  WORD_PATTERN[3] = %X'0F0F';
; 1645 3  WORD_PATTERN[4] = %X'00FF';
; 1646 3
; 1647 2  END;
; 1648 2
; 1649 2  CODECOMMENT
; 1650 2  ''
; 1651 2  'Set up the UBI and UET for a DATI, and set',
; 1652 2  'up the CMI source location (longword aligned for M = 0, word aligned',
; 1653 2  'for M = 1) with the appropriate data pattern. Execute the DATI and',
; 1654 2  'check that the UET data register contains the data pattern.',
; 1655 2  'Repeat 10 times, once for each of the five data patterns with the two',
; 1656 2  'variations of alignment.',
; 1657 2  ''
; 1658 3  BEGIN
; 1659 3
; 1660 3  INCR M FROM 0 TO 1 DO
; 1661 4  BEGIN
; 1662 4
; 1663 4  INCR N FROM 0 TO 4 DO
; 1664 5  BEGIN
; 1665 5
; 1666 5  DO ! Scope loop starts here
; 1667 6  BEGIN
; 1668 6
; 1669 6  CODECOMMENT
; 1670 6  ''
; 1671 6  'Set up the UET address register with the Unibus',
; 1672 6  'address. Bits 9 to 15 (the Unibus page frame) contain the',
; 1673 6  '7 LSBs of the UBI map number. Bits 0 to 8 contain the byte',
; 1674 6  'number field of the CMI address.',
; 1675 6  ''
; 1676 7  BEGIN
; 1677 7
; 1678 7  SCRATCH[M] = .WORD_PATTERN[N];
; 1679 7  UET_DATA_REG = 0;
; 1680 7  UBUS_PF = .MAP_NUMBER;
; 1681 7  UBUS_BO = SCRATCH[M];
; 1682 7  UET_ADDR_REG = .UBUS_ADDR;
; 1683 7
; 1684 6  END;
; 1685 6
; 1686 6  CODECOMMENT
; 1687 6  ''
; 1688 6  'Write the two MSBs of the Unibus address with the two MSBs',
; 1689 6  'of the map number.',
; 1690 6  ''

```



```

: 1691 7      BEGIN
: 1692 7
: 1693 7      TEMP3 = .MAP_NUMBER<7,2> * 8;
: 1694 7
: 1695 7
: 1696 6      END;
: 1697 6
: 1698 6      CODECOMMENT
: 1699 6      ''
: 1700 6      'Set up the UBI map. Put the CMI page frame number (PFN)',
: 1701 6      'into the register BUF_PFN. Then call the map setup routine',
: 1702 6      'with the parameters map number, PFN of the CMI buffer, byte',
: 1703 6      'offset mode (0 = no address offset), and data path number',
: 1704 6      '(0 is DDP).',
: 1705 6      '';
: 1706 7      BEGIN
: 1707 7
: 1708 7      TEMP = SCRATCH[.M];
: 1709 7      BUF_PFN = .TEMP<9,15>;
: 1710 7      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,0);
: 1711 7
: 1712 6      END;
: 1713 6
: 1714 6      CODECOMMENT
: 1715 6      ''
: 1716 6      'Execute the DATI and after a nominal wait compare the data',
: 1717 6      'in the UET data register with the expected data,',
: 1718 6      'which is contained in the table WORD_PATTERN. If there is',
: 1719 6      'an error, report it.',
: 1720 6      '';
: 1721 7      BEGIN
: 1722 7
: 1723 7      UET_CTRL_REG = .TEMP3 OR DATI;
: 1724 7
: 1725 7      $DS_WAITUS(TIME=10);
: L 1726 8      UET_STATUS(UBIMOD_DDP);          ! M7
: ; %PRINT:      Test 10, Subtest 0, Error 1
: ; 1727 7      TEMP1 = .UET_DATA_REG;
: ; 1728 7      IF .TEMP1 NEQ .WORD_PATTERN[.N]
: ; 1729 7      THEN
: ; 1730 8      BEGIN
: ; P 1731 8      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
: ; P 1732 8      PRLINK = PRINT_GENERAL,
: ; P 1733 8      P1 = FMT_DDP_DATI,P2 = 2,
: ; L 1734 8      P3 = .WORD_PATTERN[.N],P4 = .TEMP1);
: ; %PRINT:      Test 10, Subtest 0, Error 2
: ; 1735 7      END;
: ; 1736 7
: ; 1737 6      END;
: ; 1738 6
: ; 1739 6      END
: ; 1740 5      WHILE $DS_INLOOP;    ! Scope loop ends here
: ; 1741 5
: ; 1742 4      END;
: ; 1743 4

```

ZZ-ECCBA-1.4
UBI_TESTS_6_10
01-04

TEST_010: Direct Data Path DATI Test

TEST_010: Direct Data Path DATI Test

K 3

6-Sep-1985

Fiche 2 Frame K3

Sequence 242

6-Sep-1985 17:19:10

VAX-11 Bliss-32 V4.1-746

Page 59

6-Sep-1985 17:14:53

[LEMIEUX.ECCBA.RELEASE]ECCBA4.B32;6

(7)

; 1744 3
; 1745 3
; 1746 2
; 1747 2
; 1748 1
END;
END;
\$DS_ENDTEST;

00000000' 00000000' 00000000' 00000000' 00060 \$:
00000000 00000003 00070

.PSECT DISPATCH,NOWRT,NOEXE,2

.ADDRESS P.AAM, P.AAN, P.AAO, TEST_010
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

61 50 20 61 74 61 44 20 74 63 65 72 69 44 1A 000A 00080 P.AAM: .WORD 10
74 73 65 54 20 49 54 41 44 20 68 74 00082 P.AAN: .ASCII <26>\Direct Data Path DATI Test\
00091
0009D
00000000 000A0 P.AAO: .BLKB 3
.LONG 0

.PSECT TEST_010,NOWRT,2

OFFC 00000

.ENTRY TEST_010, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1581
R11
MOVAB UNIBUS_SPACE, R11
MOVAB WORD_PATTERN, R10

0000G CF 0000G 00 FB 0000C
53 0000G CF 3C 00011

CALLS #0, UNIBUS_SIZER ; 1630
MOVZWL FREE_MAP, MAP_NUMBER ; 1631

04 6A AAAA5555 8F D0 00016
08 AA 0F0F3333 8F D0 0001D
AA FF 8F 9B 00025

MOVL #-1431677611, WORD_PATTERN ; 1641
MOVL #252654387, WORD_PATTERN+4 ; 1643
MOVZBW #255, WORD_PATTERN+8 ; 1645

; Set up the UBI and UET for a DATI, and set
; up the CMI source location (longword aligned for M = 0, word
; aligned
; for M = 1) with the appropriate data pattern. Execute the DAT
; I and
; check that the UET data register contains the data pattern.
; Repeat 10 times, once for each of the five data patterns with
; the two
; variations of alignment.

```

;
59 D4 0002A CLRL M ; 1660
58 D4 0002C 1$: CLRL N ; 1663
;
; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain the
; 7 LSBs of the UBI map number. Bits 0 to 8 contain the byte
; number field of the CMI address.
;
50 0000GCF49 6A48 B0 0002E 2$: MOVW WORD_PATTERN[N], SCRATCH[M] ; 1678
6B 0003F462 8F C9 00035 BISL3 #259170, UNIBUS_SPACE, R0 ;
07 09 60 B4 0003D CLRW (R0) ; 1679
57 53 F0 0003F INSV MAP_NUMBER, #9, #7, UNIBUS_ADDR ; 1680
50 0000GCF49 3E 00044 MOVAW SCRATCH[M], R0 ; 1681
57 09 50 F0 0004A INSV R0, #0, #9, UNIBUS_ADDR ;
50 6B 0003F460 8F C9 0004F BISL3 #259168, UNIBUS_SPACE, R0 ;
60 57 B0 00057 MOVW UNIBUS_ADDR, (R0) ; 1682
;
; Write the two MSBs of the Unibus address with the two MSBs
; of the map number.
;
56 53 02 07 EF 0005A EXTZV #7, #2, MAP_NUMBER, TEMP3 ; 1693
56 56 08 C4 0005F MULL2 #8, TEMP3 ;
;
; Set up the UBI map. Put the CMI page frame number (PFN)
; into the register BUF_PFN. Then call the map setup routine
; with the parameters map number, PFN of the CMI buffer, byte
; offset mode (0 = no address offset), and data path number
; (0 is DDP).
;
52 54 54 0000GCF49 3E 00062 MOVAW SCRATCH[M], TEMP ; 1708
0F 09 EF 00068 EXTZV #9, #15, TEMP, BUF_PFN ; 1709
7E 7C 0006D CLRQ -(SP) ; 1710
52 DD 0006F PUSHL BUF_PFN ;
53 DD 00071 PUSHL MAP_NUMBER ;
0000G CF 04 FB 00073 CALLS #4, MAP_SETUP ;
;
; Execute the DATI and after a nominal wait compare the data
; in the UET data register with the expected data,
; which is contained in the table WORD_PATTERN. If there is
; an error, report it.
;
50 6B 0003F464 8F C9 00078 BISL3 #259172, UNIBUS_SPACE, R0 ; 1721
60 56 01 A9 00080 BISW3 #1, TEMP3, (R0) ; 1723
7E 0A 7D 00084 MOVQ #10, -(SP) ; 1725
00000000G 9F 02 FB 00087 CALLS #2, @DS$WAITUS ;
50 6B 0003F464 8F C9 0008E BISL3 #259172, UNIBUS_SPACE, R0 ; 1726
F8 AA 60 B0 00096 MOVW (R0), STATUS0 ;
F8 AA FF1E 8F AA 0009A BICW2 #65310, STATUS0 ;
50 F8 AA 32 000A0 CVTWL STATUS0, R0 ;
21 13 000A4 BEQL 3$ ;
7E 7C 000A6 CLRQ -(SP) ;
50 DD 000A8 PUSHL R0 ;
7E 02 7D 000AA MOVQ #2, -(SP) ;
0000G CF 9F 000AD PUSHAB FMT_UET_STAT ;

```


ZZ-ECCBA-1.4
UBI_TESTS_6_10
01-04

TEST_010: Direct Data Path DATI Test
TEST_010: Direct Data Path DATI Test

M 3

6-Sep-1985

Fiche 2 Frame M3

Sequence 244

6-Sep-1985 17:19:10

VAX-11 Bliss-32 V4.1-746

Page 61

6-Sep-1985 17:14:53

[LEMIEUX.ECCBA.RELEASE]ECCBA4.B32;6

(7)

		0000G	CF	9F	000B1	PUSHAB	PRINT_GENERAL	:
		0000G	CF	9F	000B5	PUSHAB	UBIMOD_DDP	:
	7E	0000G	CF	9A	000B9	MOVZBL	LUN, -(SP)	:
			01	DD	000BE	PUSHL	#1	:
50	00000000G	9F	0A	FB	000C0	CALLS	#10, a#DS\$ERRHARD	:
	6B	0003F462	8F	C9	000C7	BISL3	#259170, UNIBUS_SPACE, R0	: 1727
	55		60	B0	000CF	MOVW	(R0), TEMP1	:
	6A48		55	B1	000D2	CMPW	TEMP1, WORD_PATTERN[N]	: 1728
			25	13	000D6	BEQL	4\$	:
			7E	7C	000D8	CLRQ	-(SP)	: 1734
	7E		55	3C	000DA	MOVZWL	TEMP1, -(SP)	:
	7E		6A48	3C	000DD	MOVZWL	WORD_PATTERN[N], -(SP)	:
			02	DD	000E1	PUSHL	#2	:
		0000G	CF	9F	000E3	PUSHAB	FMT_DDP_DATI	:
		0000G	CF	9F	000E7	PUSHAB	PRINT_GENERAL	:
		0000G	CF	9F	000EB	PUSHAB	UBIMOD_DDP	:
	7E	0000G	CF	9A	000EF	MOVZBL	LUN, -(SP)	:
			02	DD	000F4	PUSHL	#2	:
	00000000G	9F	0A	FB	000F6	CALLS	#10, a#DS\$ERRHARD	:
	00000000G	9F	00	FB	000FD	CALLS	#0, a#DS\$INLOOP	: 1740
	03		50	E9	00104	BLBC	R0, 6\$	:
			FF24	31	00107	BRW	2\$	:
	F9	58	04	F3	0010A	AOBLEQ	#4, N, 5\$	: 1663
FF18	59	01	01	F1	0010E	ACBL	#1, #1, M, 1\$	: 1660
	00000000G	9F	00	FB	00114	CALLS	#0, a#DS\$BREAK	: 1746
	50		01	D0	0011B	MOVL	#1, R0	: 1748
			04	0011E	RET			:

; Routine Size: 287 bytes, Routine Base: TEST_010 + 0000

; 1749 1 \$DS_ENDMOD;
; 1750 1 END
; 1751 0 ELUDOM

.PSECT \$TSTCNT,NOWRT,NOEXE, OVR,2
0000000A 00000 L_TSTCNT:
.LONG 10

PSECT SUMMARY									
Name	Bytes	Attributes							
\$OWN\$	22	NOVEC,	WRT,	RD	NOEXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)	
\$PLIT\$	164	NOVEC,NOWRT,	RD	NOEXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)		
DISPATCH	120	NOVEC,NOWRT,	RD	NOEXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)		
TEST_006	632	NOVEC,NOWRT,	RD	EXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)		
TEST_007	1003	NOVEC,NOWRT,	RD	EXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)		
TEST_008	303	NOVEC,NOWRT,	RD	EXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)		
TEST_009	994	NOVEC,NOWRT,	RD	EXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)		
TEST_010	287	NOVEC,NOWRT,	RD	EXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)		
\$TSTCNT	4	NOVEC,NOWRT,	RD	NOEXE,NOSHR,	LCL,	REL,	OVR,NOPIC,ALIGN(2)		

Library Statistics						
File	-----		Symbols	-----		Processing Time
	Total	Loaded	Percent	Pages Mapped		
SYS\$COMMON:[SYSLIB]STARLET.L32;3	10093	1	0	598		00:00.8
SYS\$COMMON:[SYSLIB]DIAG.L32;265	784	20	2	85		00:00.3

COMMAND QUALIFIERS
BLISS/LIST ECCBA4.B32
Size: 3219 code + 310 data bytes
Run Time: 00:35.7
Elapsed Time: 00:42.7
Lines/CPU Min: 2944
Lexemes/CPU-Min: 32868
Memory Used: 356 pages

ZZ-ECCBA-1.4 TEST_010: Direct Data Path DATI Test
UBI TESTS_6_10
01-04 TEST_010: Direct Data Path DATI Test

; Compilation Complete

B 4

6-Sep-1985

Fiche 2 Frame B4

Sequence 246

6-Sep-1985 17:19:10

VAX-11 Bliss-32 V4.1-746

Page 63


```

; 0001 0  MODULE UBI_TESTS_11_15 (      ! UBI diagnostic program tests 11 to 15
; 0002 0      IDENT = '01-04'
; 0003 0      ) =
; 0004 1  BEGIN
; 0005 1
; 0006 1  !
; 0007 1  ! COPYRIGHT (C) 1980
; 0008 1  ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
; 0009 1  !
; 0010 1  ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
; 0011 1  ! COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
; 0012 1  ! ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
; 0013 1  ! MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
; 0014 1  ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
; 0015 1  ! TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
; 0016 1  ! REMAIN IN DEC.
; 0017 1  !
; 0018 1  ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
; 0019 1  ! AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
; 0020 1  ! CORPORATION.
; 0021 1  !
; 0022 1  ! DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
; 0023 1  ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
; 0024 1  !
; 0025 1  !
; 0026 1  !++
; 0027 1  ! FACILITY:      VAX-11 Diagnostic Library
; 0028 1  !
; 0029 1  ! ABSTRACT:      This module contains tests 11 to 15 of the COMET
; 0030 1  !                  Unibus Interface Diagnostic Program.
; 0031 1  !
; 0032 1  ! ENVIRONMENT:    VAX-11 Diagnostic Supervisor
; 0033 1  !
; 0034 1  ! AUTHOR:         Rand Gray, CREATION DATE: 11-Dec-78
; 0035 1  !
; 0036 1  ! MODIFIED BY:    Don Monroe : May 1979!
; 0037 1  !
; 0038 1  !      0-7      May 1979
; 0039 1  !                  Changed the format of the UBI MAP registers.
; 0040 1  !
; 0041 1  !      0-8      July 1979
; 0042 1  !                  Modified to support the real UET module.
; 0043 1  !
; 0044 1  !                  Fixed a BUG in XFER_SETUP routine. Loop that setup the maps
; 0045 1  !                  would not work if MAP_NUM was not zero.
; 0046 1  !
; 0047 1  !      0-9      July 1979
; 0048 1  !                  Changed references to 'Byte Offset' to 'Byte Number'
; 0049 1  !
; 0050 1  !
; 0051 1  !      0-10     October 1979
; 0052 1  !                  Added a return value to the MAP_BUFFERS routine to support
; 0053 1  !                  the changes to Module 7 Revision 10.
; 0054 1  !                  Added byte-offset parameter to MAP_BUFFERS routine so routine
; 0055 1  !                  returns with one additional page per buffer.

```

```
: 0056 1 |
: 0057 1 |
: 0058 1 | 0-11 October 22,1979
: 0059 1 | Added a routine to print MAP errors. Called via PRINT LINK
: 0060 1 | in error hard calls.
: 0061 1 |
: 0062 1 | 0-12 December 7,1979
: 0063 1 | Fixed bug in PROBE_ADDRESS routine in the ADAWI builtin.
: 0064 1 |
: 0065 1 | 0-13 April 7,1980
: 0066 1 | Changed address of unibus space so second UBI will work.
: 0067 1 |
: 0068 1 | 0-14 April 8,1980
: 0069 1 | Added a print general routine and converted PRINTX after
: 0070 1 | error calls to print links.
: 0071 1 | 1-00 June 16,1980
: 0072 1 | Initial Release
: 0073 1 | 1-01 July 31,1980
: 0074 1 | Fixed bug in MAP_BUFFERS routine that would not detect
: 0075 1 | insufficient memory for the requested number of buffers.
: 0076 1 | 1-02 December 29,1980
: 0077 1 | Fixed bug in MAP_BUFFERS routine. Supervisor allocates memory
: 0078 1 | even if not enough for requested buffer size.
: 0079 1 | 1-03 May 6, 1981
: 0080 1 | Fixed bug in map_buffers routine.
: 0081 1 |
: 0082 1 | Kevin LeMieux
: 0083 1 |
: 0084 1 | 1-04 February,1985
: 0085 1 | Fixed bug in INIT code. If two DW's were specified; after the
: 0086 1 | first pass one of them one of them was no longer tested.
: 0087 1 | Fixed minor bugs in TEST 29.
: |
: |
```

```

; 0088 1 !
; 0089 1 ! TABLE OF CONTENTS:
; 0090 1 !
; 0091 1 !   Direct Data Path DATIP/DATO Test
; 0092 1 !   Direct Data Path DATOB Test
; 0093 1 !   Buffered Address Register Test
; 0094 1 !   Buffered Data Path DATI Test
; 0095 1 !   Buffered Data Path DATIP Test
; 0096 1 !
; 0097 1 !
; 0098 1 !
; 0099 1 ! INCLUDE FILES:
; 0100 1 !
; 0101 1 !
; 0102 1 LIBRARY 'SYS$LIBRARY:STARLET';
; 0103 1 LIBRARY 'SYS$LIBRARY:DIAG';
; 0104 1 !
; 0105 1 !
; 0106 1 ! MACROS:
; 0107 1 !
; 0108 1 !
; 0109 1 !
; 0110 1 ! EQUATED SYMBOLS:
; 0111 1 !
; 0112 1 !
; 0113 1 LITERAL
; 0114 1     ALL_ONES = %X'FFFFFFFF',
; 0115 1     ALL_ZEROS = 0,
; 0116 1     BR4 = 3,
; 0117 1     BR5 = 5,
; 0118 1     BR6 = 9,
; 0119 1     BR7 = 17,
; 0120 1     CSR_MASK = %X'E0000001',
; 0121 1
; 0122 1     DATI = 1,
; 0123 1     DATIP = 3,
; 0124 1     DATO = 5,
; 0125 1     DATOB = 7,
; 0126 1
; 0127 1     DIAG_CHK_MODE = %X'C000000',
; 0128 1     DISABLE_ECC = %X'2000000',
; 0129 1     ENABLE_ECC = %X'2000000',
; 0130 1     NORMAL = %X'E000000',
; 0131 1     ERR = BIT31,
; 0132 1     MAP_MASK = %X'82607FFF',
; 0133 1     MEMORY_CONT = %X'F20000',
; 0134 1     NON_XIST_NEXUS = %X'F40000',
; 0135 1     NXM = BIT30,
; 0136 1     PURGE = BIT0,
; 0137 1     UCE = BIT29;
; 0138 1 !
; 0139 1 ! OWN STORAGE:
; 0140 1 !
; 0141 1 ! OWN
; 0142 1     BYTE_PATTERN: VECTOR[4,WORD], ! Data patterns loaded at run time

```

! M7


```

; 0143 1 STATUS0: SIGNED WORD, ! Used by the UET status macro
; 0144 1 STATUS1, ! Used by the UBI status macro
; 0145 1 WORD_PATTERN: VECTOR[5,WORD]; ! Data patterns loaded at run time
; 0146 1
; 0147 1 !
; 0148 1 ! EXTERNAL REFERENCES:
; 0149 1 !
; 0150 1 EXTERNAL ROUTINE
; 0151 1 DECODER,
; 0152 1 ENCODER,
; 0153 1 MAP_SETUP,
; 0154 1 PRINT_GENERAL:NOVALUE,
; 0155 1 PROBE_ADDRESS,
; 0156 1 UNIBUS_SIZER,
; 0157 1 XFER_SETUP;
; 0158 1
; 0159 1 EXTERNAL
; 0160 1 BUFFER_SETUP,
; 0161 1 CODEWORDS: VECTOR[9,WORD],
; 0162 1 UBI_UNDER_TEST,
; 0163 1 UNIBUS_SPACE,
; 0164 1 DECODED_WORDS: VECTOR[9],
; 0165 1 EVENT_FLAG: BITVECTOR[4],
; 0166 1 FMT_BDP_DATI,
; 0167 1 FMT_BDP_DATO,
; 0168 1 FMT_BDP_DATO,
; 0169 1 FMT_BYTE_OFF,
; 0170 1 FMT_CMI_UB,
; 0171 1 FMT_CPU_RW,
; 0172 1 FMT_DDP_DATI,
; 0173 1 FMT_DDP_DATO,
; 0174 1 FMT_DDP_DATO,
; 0175 1 FMT_DP_SEL,
; 0176 1 FMT_INTERLOCK, ! A10
; 0177 1 FMT_MAP_ADDR, ! D13
; 0178 1 ! FMT_MAP_BIT,
; 0179 1 FMT_MAP_CS, ! M8
; 0180 1 FMT_MAP_DB_SA0, ! A8
; 0181 1 FMT_MAP_DB_SA1,
; 0182 1 FMT_MAP_FAIL,
; 0183 1 FMT_MCHK,
; 0184 1 FMT-UA1,
; 0185 1 FMT-UB_CMI,
; 0186 1 FMT-UET_STAT,
; 0187 1 FREE_MAP: VECTOR[4,WORD],
; 0188 1 HANDLER,
; 0189 1 INTERRUPT_EXP: BYTE,
; 0190 1 INTERRUPT_OCC: BYTE,
; 0191 1 LUN: BYTE,
; 0192 1 MAP_BUFFERS,
; 0193 1 NOISY_WORDS: VECTOR[9,WORD],
; 0194 1 NUM_UB,
; 0195 1 PRINT_CSR_ERR,
; 0196 1 PRINT_DCMF_ERR,
; 0197 1 UBIMOD, ! A10

```

```

; 0198 1 UBIMOD_BDP,
; 0199 1 UBIMOD_BYTE_OFF,
; 0200 1 UBIMOD_CMI,
; 0201 1 UBIMOD_CPU_RW,
; 0202 1 UBIMOD_CSR,
; 0203 1 UBIMOD_DDP,
; 0204 1 UBIMOD_DP_SEL,
; 0205 1 UBIMOD_MAP,
; 0206 1 UBIMOD_MAP_CS,
; 0207 1 UBIMOD_MAP_ADDR,
; 0208 1 UBIMOD_MAP_CTRL,
; 0209 1 UBIMOD_UA1,
; 0210 1 UBIMOD_UB_CMI,
; 0211 1 SCRATCH: VECTOR[128],
; 0212 1 UBE_BASE_ADDR: VECTOR[4],
; 0213 1 UBE_BUF_SIZE,
; 0214 1 ! UBE_ERROR, ! D8
; 0215 1 UBE_STAT_ERR: BITVECTOR[4],
; 0216 1 UBE0_ISR,
; 0217 1 UET_ISR,
; 0218 1 VALID_MAPS: BITVECTOR[512];
; 0219 1
; 0220 1 MACRO
; 0221 1 !++
; 0222 1 ! This macro defines the address of the control and status register
; 0223 1 ! for the given buffered data path number.
; 0224 1 !--
; 0225 1 UBI_CSR(BDP) = (.UBI_UNDER_TEST + BDP*4)%;
; 0226 1
; 0227 1 !++
; 0228 1 ! This macro defines the address of a UBI map for the given map
; 0229 1 ! number.
; 0230 1 !--
; 0231 1 UBI_MAP(NUM) = (.UBI_UNDER_TEST + %X'800' + NUM*4)%;
; 0232 1
; 0233 1 !++
; 0234 1 ! These macros temporarily define the addresses of the (simulated) UET
; 0235 1 ! registers. They will be replaced for the real UET.
; 0236 1 !--
; 0237 1 MACRO ! A8
; 0238 1 UET_ADDR_REG = (.UNIBUS_SPACE OR %X'3F460')<0,16>%; ! A8 M14
; 0239 1 UET_DATA_REG = (.UNIBUS_SPACE OR %X'3F462')<0,16>%; ! A8 M14
; 0240 1 UET_DATA_REGB = (.UNIBUS_SPACE OR %X'3F462')<0,8>%; ! A8 M14
; 0241 1 UET_CTRL_REG = (.UNIBUS_SPACE OR %X'3F464')<0,16>%; ! A8 M14
; 0242 1
; 0243 1 !++ ! A8
; 0244 1 ! This macro is used for checking the status of the UET. ! A8
; 0245 1 !-- ! A8
; M 0246 1 UET_STATUS(CALLOUT) = ! A8
; M 0247 1 STATUS0 = .UET_CTRL_REG; ! Read the UET control register ! A8
; M 0248 1 STATUS0 = .STATUS0 AND %X'E1'; ! A8
; M 0249 1 IF .STATUS0 NEQ 0 ! A8
; M 0250 1 THEN ! A8
; M 0251 1 BEGIN ! A8
; M 0252 1 $DS_ERRHARD(UNIT=.LUN,MSGADR=CALLOUT,

```

```

; M 0253 1 PRLINK = PRINT_GENERAL,
; M 0254 1 P1 = FMT_UET_STAT,P2 = 2,
; M 0255 1 P3 = 0,P4 = .STATUS0); ! A8
; 0256 1 ! A8
; 0257 1 ! A8
; 0258 1
; 0259 1 MACRO
; 0260 1 !++
; 0261 1 ! This macro is used for checking the status of the UBI.
; 0262 1 !--
; M 0263 1 UBI_STATUS(BDP,CALLOUT) =
; M 0264 1 STATUS1 = .UBI_CSR(BDP) AND CSR_MASK;
; M 0265 1 IF .STATUS1 LSS 0
; M 0266 1 THEN
; M 0267 1 $DS_ERRHARD(UNIT=.LUN,
; 0268 1 MSGADR=CALLOUT,
; 0269 1 PRLINK=PRINT_CSR_ERR,
; 0270 1 P1=BDP,P2=0,P3=.STATUS1);x;
; 0271 1
; 0272 1 !
; L 0273 1 $DS_BGNMOD(ENV=0,TEST=11);
; *PRINT: 1 Bliss-32 Diagnostic Macro Library version 7.0 "DIAG.L32 (57)"
; 0274 1 $DS_DSADef;
; 0275 1 $DS_SECDEF(ALL,UBE);
; 0276 1
; 0277 1 BUILTIN
; 0278 1 ROT; ! This is so we can do ROTLs

```


TEST_011: Direct Data Path DATIP/DATO Test

\$DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Direct Data Path DATIP/DATO Test');

!++
! TEST DESCRIPTION:

This tests the ability of the UBI to perform a DATIP/DATO transaction through the direct data path (DDP). The transactions are made from longword and word aligned CMI addresses. The following data patterns are used:

pattern 1	0101010101010101
pattern 2	1010101010101010
pattern 3	0011001100110011
pattern 4	0000111100001111
pattern 5	0000000011111111

Before testing starts, the Unibus Sizer routine is called to determine which maps are useable.

Subtest 2 checks that a DATIP from the UET locks the CMI from other READLOCK transactions. This is performed by causing the UET to execute the DATIP then performing an INTERLOCK READ from the CPU. The CPU reference should cause a machine check.

! ASSUMPTIONS:

! Test 1-Test 7

! REGISTER

! BUF_PFN,
! M,
! MAP_NUMBER,
! N,
! TEMP,
! TEMP0:WORD,
! TEMP1,
! TEMP3,
! TEMP4,
! UBUS_ADDR: WORD;

! BUILTIN

! MTPR;

! MACRO

! UBUS_PF = UBUS_ADDR<9,7>X,
! UBUS_BO = UBUS_ADDR<0,9>X;

! Defines the Unibus page frame field
! Defines the Unibus byte number field

! MAP

! SCRATCH: VECTOR[5,WORD];

! CODECOMMENT

! 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
! 'a table of useable map entries. In the test which follows, the map',

```

: 0334 2 'selected is drawn from this table.',
: 0335 2 '':
: 0336 3 BEGIN
: 0337 3
: 0338 3 UNIBUS_SIZER();
: 0339 3 MAP_NUMBER = .FREE_MAP[0];
: 0340 3
: 0341 2 END;
: 0342 2
: 0343 2 CODECOMMENT
: 0344 2 '':
: 0345 2 'Load the data patterns into the table WORD_PATTERN for later use.',
: 0346 2 '':
: 0347 3 BEGIN
: 0348 3
: 0349 3 WORD_PATTERN[0] = %X'5555';
: 0350 3 WORD_PATTERN[1] = %X'AAAA';
: 0351 3 WORD_PATTERN[2] = %X'3333';
: 0352 3 WORD_PATTERN[3] = %X'0F0F';
: 0353 3 WORD_PATTERN[4] = %X'00FF';
: 0354 3
: 0355 2 END;
: 0356 2
: 0357 3 $DS BGNSUB;
: 0358 3 CODECOMMENT
: 0359 3 '':
: 0360 3 'Set up the UBI and UET for a DATIP, and',
: 0361 3 'set up the CMI source location (longword aligned for M = 0, word',
: 0362 3 'aligned for M = 1) with the appropriate data pattern. Execute the DATIP',
: 0363 3 'and check that the UET data register contains the data pattern.',
: 0364 3 'Check also that the CMI source location contains the pattern rotated',
: 0365 3 'left one bit as a result of the action of the UBE.',
: 0366 3 'Repeat 10 times, once for each of the five data patterns with the two',
: 0367 3 'variations of alignment.',
: 0368 3 '':
: 0369 4 BEGIN
: 0370 4
: 0371 4 INCR M FROM 0 TO 1 DO
: 0372 5 BEGIN
: 0373 5
: 0374 5 INCR N FROM 0 TO 4 DO
: 0375 6 BEGIN
: 0376 6
: 0377 6 DO ! Scope loop begins here
: 0378 7 BEGIN
: 0379 7
: 0380 7 CODECOMMENT
: 0381 7 '':
: 0382 7 'Set up the UET address register with the Unibus',
: 0383 7 'address. Bits 9 to 15 (the Unibus page frame) contain the',
: 0384 7 '7 LSBs of the UBI map number. Bits 0 to 8 contain the byte',
: 0385 7 'number field of the CMI address.',
: 0386 7 '':
: 0387 8 BEGIN
: 0388 8

```

```

; 0389 8          SCRATCH(.M) = .WORD_PATTERN(.N);
; 0390 8          UET_DATA_REG = 0;
; 0391 8          UBUS_PF = .MAP_NUMBER;
; 0392 8          UBUS_BO = SCRATCH(.M);
; 0393 8          UET_ADDR_REG = .UBUS_ADDR;
; 0394 8
; 0395 7          END;
; 0396 7
; 0397 7          CODECOMMENT
; 0398 7          ''
; 0399 7          'Write the two MSBs of the Unibus address with the two MSBs',
; 0400 7          'of the map number.',
; 0401 7          ''
; 0402 8          BEGIN
; 0403 8
; 0404 8              TEMP3 = .MAP_NUMBER<7,2> * 8;
; 0405 8
; 0406 7          END;
; 0407 7
; 0408 7          CODECOMMENT
; 0409 7          ''
; 0410 7          'Set up the UBI map. Put the CMI page frame number (PFN)',
; 0411 7          'into the register BUF_PFN. Then call the map setup routine',
; 0412 7          'with the parameters map number, PFN of the CMI buffer, byte',
; 0413 7          'offset mode (0 = no address offset), and data path number',
; 0414 7          '(0 is DDP).',
; 0415 7          ''
; 0416 8          BEGIN
; 0417 8
; 0418 8              TEMP = SCRATCH(.M);
; 0419 8              BUF_PFN = .TEMP<9,15>;
; 0420 8              MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,0);
; 0421 8
; 0422 7          END;
; 0423 7
; 0424 7          CODECOMMENT
; 0425 7          ''
; 0426 7          'Execute the DATIP and after a nominal wait compare the data',
; 0427 7          'in the UET data register with the expected data,',
; 0428 7          'which is contained in the table WORD_PATTERN. If there is',
; 0429 7          'an error, report it.',
; 0430 7          ''
; 0431 8          BEGIN
; 0432 8
; 0433 8              UET_CTRL_REG = .TEMP3 OR DATIP;
; 0434 8
; 0435 8              $DS_WAITUS(TIME=10);
; 0436 9          L 0436 9          UET_STATUS(UBIMOD_DDP);
; 0437 8          xPRINT: Test 11, Subtest 1, Error 1
; 0438 8              TEMP = .UET_DATA_REG;
; 0439 8              TEMPO = .WORD_PATTERN(.N);
; 0440 8              IF .TEMP NEQ .TEMPO
; 0441 9              THEN
; 0442 9                  BEGIN
; 0442 9                      TEMP4 = -1;

```

! M7


```
; 0443 9      MTPR(TEMP4, %X'37');
; P 0444 9      $DS_ERRHARD(UNIT=.LUN, MSGADR=UBIMOD_DDP,
; P 0445 9      PRLINK = PRINT_GENERAL,
; P 0446 9      P1 = FMT_DDP_DATI, P2 = 2,
; L 0447 9      P3 = .WORD_PATTERN[.N], P4 = .TEMP);
; %PRINT:      Test 11, Subtest 1, Error 2
; 0448 8      END;
; 0449 8
; 0450 8      CODECOMMENT
; 0451 8      '
; 0452 8      'Rotate the data in the data register ( and the',
; 0453 8      'expected data) and perform a DATO.',
; 0454 8      '
; 0455 9      BEGIN
; 0456 9      TEMP1 = .UET_DATA_REG;
; 0457 9      TEMP1<16,16> = .TEMP1<0,16>;
; 0458 9      TEMP1 = ROT(.TEMP1,1);
; 0459 9      UET_DATA_REG = .TEMP1;
; 0460 9      TEMP1 = .WORD_PATTERN[.N];
; 0461 9      TEMP1<16,16> = .WORD_PATTERN[.N];
; 0462 9      TEMP1 = ROT(.TEMP1,1);
; 0463 9      TEMP0 = .TEMP1;
; 0464 9      UET_CTRL_REG = .TEMP3 OR DATO;
; 0465 9      $DS_WAITUS(TIME=10);
; L 0466 10     UET_STATUS(UBIMOD_DDP);
; %PRINT:      Test 11, Subtest 1, Error 3
; 0467 8      END;
; 0468 7      END;
; 0469 7
; 0470 7      CODECOMMENT
; 0471 7      '
; 0472 7      'Now see if the DATO part worked. The UET data',
; 0473 7      'register contents are rotated left one bit when a DATO is',
; 0474 7      'issued following a DATIP. Here the data pattern shifted',
; 0475 7      'left one bit is compared to the CMI destination, and any',
; 0476 7      'errors are reported.',
; 0477 7      '
; 0478 8      BEGIN
; 0479 8
; 0480 8      IF .SCRATCH[.M] NEQ .TEMP0
; 0481 8      THEN
; 0482 9      BEGIN
; 0483 9      TEMP4 = -1;
; 0484 9      MTPR(TEMP4, %X'37');
; P 0485 9      $DS_ERRHARD(UNIT=.LUN, MSGADR=UBIMOD_DDP,
; P 0486 9      PRLINK = PRINT_GENERAL,
; P 0487 9      P1 = FMT_DDP_DATO, P2 = 3,
; P 0488 9      P3 = SCRATCH[.M], P4 = .TEMP,
; L 0489 9      P5 = .SCRATCH[.M]); ! M12
; %PRINT:      Test 11, Subtest 1, Error 4
; 0490 8      END;
; 0491 8
; 0492 7      END;
; 0493 7
; 0494 7      END
```

```

: 0495 6      WHILE $DS_INLOOP;    ! Scope loop ends here
: 0496 6
: 0497 5      END;
: 0498 5
: 0499 4      END;
: 0500 4
: 0501 3      END;
: 0502 3
: 0503 2      $DS_ENDSUB;
: 0504 2
: 0505 3      $DS_BGNSUB;
: 0506 3      DO                      ! Scope loop starts here
: 0507 4      BEGIN
: 0508 4      CODECOMMENT
: 0509 4      ''
: 0510 4      'This subtest was added in version 1-11. It checks that the CMI is INTERLOCKED',
: 0511 4      'when the UET performs a DATIP.',
: 0512 4      '';
: 0513 5      BEGIN
: 0514 5      CODECOMMENT
: 0515 5      ''
: 0516 5      'Setup the UET registers.',
: 0517 5      '';
: 0518 6      BEGIN
: 0519 6      SCRATCH[0] = .WORD_PATTERN[0];
: 0520 6      UET_DATA_REG = 0;
: 0521 6      UBUS_PF = .MAP_NUMBER;
: 0522 6      UBUS_BO = SCRATCH[0];
: 0523 6      UET_ADDR_REG = .UBUS_ADDR;
: 0524 6      TEMP3 = .MAP_NUMBER<7,2> * 8;
: 0525 5      END;
: 0526 5
: 0527 5      CODECOMMENT
: 0528 5      ''
: 0529 5      'Setup the UBI map.',
: 0530 5      '';
: 0531 6      BEGIN
: 0532 6      TEMP = SCRATCH[0];
: 0533 6      BUF_PFN = .TEMP<9,15>;
: 0534 6      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,0);
: 0535 5      END;
: 0536 5
: 0537 5      CODECOMMENT
: 0538 5      ''
: 0539 5      'Execute the DATIP and after a nominal wait, execute an INTERLOCKED',
: 0540 5      'READ from the CPU and check that it machine checks.',
: 0541 5      '';
: 0542 6      BEGIN
: 0543 6      UET_CTRL_REG = .TEMP3 OR DATIP;
: 0544 6
: 0545 6      $DS_WAITUS(TIME = 10);
: 0546 6
: 0547 6      IF PROBE_ADDRESS(SCRATCH[0],0,1)
: 0548 6      THEN
: 0549 7      BEGIN

```

ZZ-ECCBA-1.4
UBI_TESTS_11_15
01-04

TEST_011: Direct Data Path DATIP/DATO Test
TEST_011: Direct Data Path DATIP/DATO Test

N 4
6-Sep-1985 Fiche 2 Frame N4 Sequence 258
6-Sep-1985 17:19:55 VAX-11 Bliss-32 V4.1-746 Page 12
6-Sep-1985 17:14:57 [LEMIEUX.ECCBA.RELEASE]ECCBA5.B32;5 (3)

```
; P 0550 7          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD,  
; P 0551 7          PRLINK = PRINT_GENERAL,  
; L 0552 7          P1 = FMT_INTERLOCK,P2 = 0);  
; xPRINT:          Test 11, Subtest 2, Error 5  
; 0553 6          END;  
; 0554 6          UET_CTRL_REG = .TEMP3 OR DATO;      ! Release the interlock  
; 0555 5          END;  
; 0556 4          END;  
; 0557 4          END  
; 0558 3          WHILE $DS_INLOOP;                  ! End of scope loop  
; 0559 2          $DS_ENDSUB;  
; 0560 2  
; 0561 1          $DS_ENDTEST;
```

.TITLE UBI_TESTS_11_15
.IDENT \01-04\

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00000 \$:
00000000 00000003 00010

.ADDRESS P.AAA, P.AAB, P.AAC, TEST_011
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

```
61 50 20 61 74 61 44 20 74 63 65 72 69 44 20 000B 00000 P.AAA: .WORD 11  
54 20 4F 54 41 44 2F 50 49 54 41 44 20 68 74 00002 P.AAB: .ASCII \ Direct Data Path DATIP/DATO Test\  
74 73 65 00011  
00020  
00023 .BLKB 1  
00000000 00024 P.AAC: .LONG 0
```

.PSECT \$OWN\$,NOEXE,2

00000 BYTE_PATTERN:
 .BLKB 8
00008 STATUS0: .BLKB 2
0000A .BLKB 2
0000C STATUS1: .BLKB 4
00010 WORD_PATTERN:
 .BLKB 10

DSA\$AL_APTMAIL= 65024
DSA\$GL_FLAGS= 65024
DSA\$GL_APTCOM= 65028
DSA\$GL_PASSES= 65032
DSA\$GL_UNITS= 65036
DSA\$GL_SECTNO= 65040
DSA\$GL_SID= 65044
DSA\$GL_MSGTYP= 65088
DSA\$GL_ERRNO= 65092
DSA\$GL_EVENT= 65096
DSA\$GL_SUBTNO= 65100
DSA\$GL_TESTNO= 65104
DSA\$GL_PASSNO= 65108

ZZ-ECCBA-1.4 TEST_011: Direct Data Path DATIP/DATO Test
UBI_TESTS_11_15
01-04 TEST_011: Direct Data Path DATIP/DATO Test

B 5
6-Sep-1985 Fiche 2 Frame B5 Sequence 259
6-Sep-1985 17:19:55 VAX-11 Bliss-32 V4.1-746 Page 13
6-Sep-1985 17:14:57 [LEMIEUX.ECCBA.RELEASE]ECCBA5.B32;5 (3)

DSA\$GL_DEVLEN= 65112
DSA\$GL_DEVNAM= 65116
DSA\$GL_MSGPTR= 65128
.EXTRN DECODER, ENCODER
.EXTRN MAP_SETUP, PRINT_GENERAL
.EXTRN PROBE_ADDRESS, UNIBUS_SIZER
.EXTRN XFER_SETUP, BUFFER_SETUP
.EXTRN CODEWORDS, UBI_UNDER_TEST
.EXTRN UNIBUS_SPACE, DECODED_WORDS
.EXTRN EVENT_FLAG, FMT_BDP_DATI
.EXTRN FMT_BDP_DATO, FMT_BDP_DATO
.EXTRN FMT_BYTE_OFF, FMT_CMI_UB
.EXTRN FMT_CPU_RW, FMT_DDP_DATI
.EXTRN FMT_DDP_DATO, FMT_DDP_DATO
.EXTRN FMT_DP_SEL, FMT_INTERLOCK
.EXTRN FMT_MAP_ADDR, FMT_MAP_CS
.EXTRN FMT_MAP_DB_SA0, FMT_MAP_DB_SA1
.EXTRN FMT_MAP_FAIL, FMT_MCHK
.EXTRN FMT_UA1, FMT_UB_CMI
.EXTRN FMT_UET_STAT, FREE_MAP
.EXTRN HANDLER, INTERRUPT_EXP
.EXTRN INTERRUPT_OCC, LUN
.EXTRN MAP_BUFFERS, NOISY_WORDS
.EXTRN NUM_UB, PRINT_CSR_ERR
.EXTRN PRINT_DCMP_ERR, UBIMOD
.EXTRN UBIMOD_BDP, UBIMOD_BYTE_OFF
.EXTRN UBIMOD_CMI, UBIMOD_CPU_RW
.EXTRN UBIMOD_CSR, UBIMOD_DDP
.EXTRN UBIMOD_DP_SEL, UBIMOD_MAP
.EXTRN UBIMOD_MAP_CS, UBIMOD_MAP_ADDR
.EXTRN UBIMOD_MAP_CTRL
.EXTRN UBIMOD_UA1, UBIMOD_UB_CMI
.EXTRN SCRATCH, UBE_BASE_ADDR
.EXTRN UBE_BUF_SIZE, UBE_STAT_ERR
.EXTRN UBE0_ISR, UET_ISR
.EXTRN VALID_MAPS, DS\$BGNSUB
.EXTRN DS\$WAITUS, DS\$ERRHARD
.EXTRN DS\$INLOOP, DS\$ENDSUB
.EXTRN DS\$BREAK

.PSECT TEST_011,NOWRT,2

OFFC 00000

.ENTRY TEST_011, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0279
R11 ;

; Call the Unibus sizer routine to locate any Unibus memory. This builds
; a table of useable map entries. In the test which follows, the
; selected is drawn from this table.

0000G CF 00 FB 00002
53 0000G CF 3C 00007

CALLS #0, UNIBUS_SIZER ; 0338
MOVZWL FREE_MAP, MAP_NUMBER ; 0339

; Load the data patterns into the table WORD_PATTERN for later

;

```

0000' CF AAAA5555 8F D0 0000C
0000' CF 0F0F3333 8F D0 00015
0000' CF          8F 9B 0001E
                   01 DD 00024
                   0B DD 00026
00000000G 9F      02 FB 00028

```

use.

```

; MOVL #-1431677611, WORD_PATTERN ; 0349
; MOVL #252654387, WORD_PATTERN+4 ; 0351
; MOVZBW #255, WORD_PATTERN+8 ; 0353
; PUSHL #1 ; 0355
; PUSHL #11 ;
; CALLS #2, @DS$BGNSUB ;

```

;

;

;

;

;

;

```

; Set up the UBI and UET for a DATIP, and
; set up the CMI source location (longword aligned for M = 0, w
ord
; aligned for M = 1) with the appropriate data pattern. Execute
the DATIP
; and check that the UET data register contains the data patter
n.
; Check also that the CMI source location contains the pattern
rotated
; left one bit as a result of the action of the UBE.
; Repeat 10 times, once for each of the five data patterns with
the two
; variations of alignment.

```

```

5A D4 0002F
5B D4 00031

```

```

; CLRL M ; 0371
; CLRL N ; 0374

```

```

; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain the
; 7 LSBs of the UBI map number. Bits 0 to 8 contain the byte
; number field of the CMI address.

```

```

50 0000GCF4A 0000'CF4B B0 00033
0000G CF 0003F462 8F C9 0003C
                   60 B4 00046
59 07 09 53 F0 00048
                   50 0000GCF4A 3E 0004D
59 09 00 50 F0 00053
50 0000G CF 0003F460 8F C9 00058
                   60 59 B0 00062

```

```

; 2$: MOVW WORD_PATTERN[N], SCRATCH[M] ; 0389
; BISL3 #259170, UNIBUS_SPACE, R0 ;
; CLRW (R0) ; 0390
; INSV MAP_NUMBER, #9, #7, UNIBUS_ADDR ; 0391
; MOVAW SCRATCH[M], R0 ; 0392
; INSV R0, #0, #9, UNIBUS_ADDR ;
; BISL3 #259168, UNIBUS_SPACE, R0 ;
; MOVW UNIBUS_ADDR, (R0) ; 0393

```

```

; Write the two MSBs of the Unibus address with the two MSBs
; of the map number.

```

```

57 53 02 07 EF 00065
57 57 08 C4 0006A

```

```

; EXTZV #7, #2, MAP_NUMBER, TEMP3 ; 0404
; MULL2 #8, TEMP3 ;

```

```

; Set up the UBI map. Put the CMI page frame number (PFN)
; into the register BUF_PFN. Then call the map setup routine
; with the parameters map number, PFN of the CMI buffer, byte
; offset mode (0 = no address offset), and data path number
; (0 is DDP).

```

```

52 54 54 0000GCF4A 3E 0006D
0F 09 EF 00073
7E 7C 00078

```

```

; MOVAW SCRATCH[M], TEMP ; 0418
; EXTZV #9, #15, TEMP, BUF_PFN ; 0419
; CLRQ -(SP) ; 0420

```

```

52 DD 0007A    PUSHL BUF_PFN
53 DD 0007C    PUSHL MAP_NUMBER
04 FB 0007E    CALLS #4, MAP_SETUP

```

```

; Execute the DATIP and after a nominal wait compare the data
; in the UET data register with the expected data,
; which is contained in the table WORD_PATTERN. If there is
; an error, report it.

```

```

50 0000G CF 0003F464 8F C9 00083    BISL3 #259172, UNIBUS_SPACE, R0 ; 0431
60 57 03 A9 0008D    BISW3 #3, TEMP3, (R0) ; 0433
7E 0A 7D 00091    MOVQ #10, -(SP) ; 0435
00000000G 9F 02 FB 00094    CALLS #2, a#DS$WAITUS ;
50 0000G CF 0003F464 8F C9 0009B    BISL3 #259172, UNIBUS_SPACE, R0 ; 0436
0000' CF 60 B0 000A5    MOVW (R0), STATUS0 ;
0000' CF FF1E 8F AA 000AA    BICW2 #65310, STATUS0 ;
50 0000' CF 32 000B1    CVTWL STATUS0, R0 ;
21 13 000B6    BEQL 3$ ;
7E 7C 000B8    CLRQ -(SP) ;
50 DD 000BA    PUSHL R0 ;
7E 02 7D 000BC    MOVQ #2, -(SP) ;
0000G CF 9F 000BF    PUSHAB FMT UET_STAT ;
0000G CF 9F 000C3    PUSHAB PRINT_GENERAL ;
0000G CF 9F 000C7    PUSHAB UBIMOD_DDP ;
7E 0000G CF 9A 000CB    MOVZBL LUN, -(SP) ;
01 DD 000D0    PUSHL #1 ;
00000000G 9F 0A FB 000D2    CALLS #10, a#DS$ERRHARD ;
50 0000G CF 0003F462 8F C9 000D9 3$: BISL3 #259170, UNIBUS_SPACE, R0 ; 0437
54 60 3C 000E3    MOVZWL (R0), TEMP ;
55 0000'CF4B B0 000E6    MOVW WORD_PATTERN[N], TEMPO ; 0438
54 10 00 ED 000EC    CMPZV #0, #16, TEMPO, TEMP ; 0439
58 2C 13 000F1    BEQL 4$ ;
37 01 CE 000F3    MNEGL #1, TEMP4 ; 0442
58 DA 000F6    MTPR TEMP4, #55 ; 0443
7E 7C 000F9    CLRQ -(SP) ; 0447
54 DD 000FB    PUSHL TEMP ;
7E 0000'CF4B 3C 000FD    MOVZWL WORD_PATTERN[N], -(SP) ;
02 DD 00103    PUSHL #2 ;
0000G CF 9F 00105    PUSHAB FMT_DDP_DATI ;
0000G CF 9F 00109    PUSHAB PRINT_GENERAL ;
0000G CF 9F 0010D    PUSHAB UBIMOD_DDP ;
7E 0000G CF 9A 00111    MOVZBL LUN, -(SP) ;
02 DD 00116    PUSHL #2 ;
00000000G 9F 0A FB 00118    CALLS #10, a#DS$ERRHARD ;

```

```

; Rotate the data in the data register ( and the
; expected data) and perform a DATO.

```

```

50 0000G CF 0003F462 8F C9 0011F 4$: BISL3 #259170, UNIBUS_SPACE, R0 ; 0456
56 60 3C 00129    MOVZWL (R0), TEMP1 ;
10 10 56 F0 0012C    INSV TEMP1, #16, #16, TEMP1 ; 0457
56 01 9C 00131    ROTL #1, TEMP1, TEMP1 ; 0458
60 56 B0 00135    MOVW TEMP1, (R0) ; 0459
56 0000'CF4B 3C 00138    MOVZWL WORD_PATTERN[N], TEMP1 ; 0460
0000'CF4B 3F 0013E    PUSHAB WORD_PATTERN[N] ; 0461

```



```

INSV      a(SP)+, #16, #16, TEMP1
ROTL      #1, TEMP1, TEMP1
MOVW      TEMP1, TEMP0
BISL3     #259172, UNIBUS_SPACE, R0
BISW3     #5, TEMP3, (R0)
MOVQ      #10, -(SP)
CALLS     #2, a#DS$WAITUS
BISL3     #259172, UNIBUS_SPACE, R0
MOVW      (R0), STATUS0
BICW2     #65310, STATUS0
CVTWL     STATUS0, R0
BEQL      5$
CLRQ      -(SP)
PUSHL     R0
MOVQ      #2, -(SP)
PUSHAB    FMT_UET_STAT
PUSHAB    PRINT_GENERAL
PUSHAB    UBIMOD_DDP
MOVZBL    LUN, -(SP)
PUSHL     #3
CALLS     #10, a#DS$ERRHARD

```

; 0462
; 0463
;
; 0464
; 0465
; 0466
;
;
;
;
;
;
;
;
;
;
;

```
; Now see if the DATO part worked. The UET data
; register contents are rotated left one bit when a DATO is
; issued following a DATIP. Here the data pattern shifted
; left one bit is compared to the CMI destination, and any
; errors are reported.
```

```

5$:  CMPW      SCRATCH[M], TEMP0
      BEQL     6$
      MNEGL    #1, TEMP4
      MTPR     TEMP4, #55
      CLRL     -(SP)
      MOVZWL    SCRATCH[M], -(SP)
      PUSHL     TEMP
      PUSHAW    SCRATCH[M]
      PUSHL     #3
      PUSHAB    FMT_DDP_DATO
      PUSHAB    PRINT_GENERAL
      PUSHAB    UBIMOD_DDP
      MOVZBL    LUN, -(SP)
      PUSHL     #4
      CALLS     #10, a#DS$ERRHARD
6$:  CALLS     #0, a#DS$INLOOP
      BLBC      R0, 8$
7$:  BRW       2$
8$:  AOBLEQ    #4, N, 7$
      ACBL     #1, #1, M, 1$
      PUSHL     #1
      PUSHL     #11
      CALLS     #2, a#DS$ENDSUB
      PUSHL     #2
      PUSHL     #11
      CALLS     #2, a#DS$BGNSUB

```

: 0480
 :
 : 0483
 : 0484
 : 0489
 :
 :
 :
 :
 :
 :
 :
 :
 :
 :
 : 0495
 :
 :
 : 0374
 : 0371
 : 0501
 :
 :
 : 0503
 :
 :

FE3C

F9
SA

```

;
; This subtest was added in version 1-11. It checks that the CM
; I is INTERLOCKED
; when the UET performs a DATIP.
;
; Setup the UET registers.
;
50      0000G CF      0000' CF B0 0020B 9$: MOVW WORD_PATTERN, SCRATCH ; 0519
      0000G CF 0003F462 8F C9 00212 BLSL3 #259170, UNIBUS_SPACE, R0 ;
59      07      09      60 B4 0021C CLRW (R0) ; 0520
      50      0000G CF 53 F0 0021E INSV MAP_NUMBER, #9, #7, UBUS_ADDR ; 0521
59      09      00      50 F0 00223 MOVAB SCRATCH, R0 ; 0522
      50      0000G CF 0003F460 8F C9 0022D INSV R0, #0, #9, UBUS_ADDR ;
57      53      02      59 B0 00237 BLSL3 #259168, UNIBUS_SPACE, R0 ;
      57      07      07 EF 0023A MOVW UBUS_ADDR, (R0) ; 0523
      57      08 C4 0023F EXTZV #7, #2, MAP_NUMBER, TEMP3 ; 0524
      57      08 C4 0023F MULL2 #8, TEMP3 ;
;
; Setup the UBI map.
;
52      54      54      0000G CF 9E 00242 MOVAB SCRATCH, TEMP ; 0532
      54      0F      09 EF 00247 EXTZV #9, #15, TEMP, BUF_PFN ; 0533
      54      0F      7E 7C 0024C CLRQ -(SP) ; 0534
      54      0F      52 DD 0024E PUSHL BUF_PFN ;
      54      0F      53 DD 00250 PUSHL MAP_NUMBER ;
      54      0F      04 FB 00252 CALLS #4, MAP_SETUP ;
;
; Execute the DATIP and after a nominal wait, execute an INTERL
; OCKED
; READ from the CPU and check that it machine checks.
;
50      0000G CF 0003F464 8F C9 00257 BLSL3 #259172, UNIBUS_SPACE, R0 ; 0542
60      57      03 A9 00261 BLSW3 #3, TEMP3, (R0) ; 0543
      7E      0A 7D 00265 MOVQ #10, -(SP) ; 0545
      00000000G 9F      02 FB 00268 CALLS #2, a#DS$WAITUS ;
      00000000G 9F      01 DD 0026F PUSHL #1 ; 0547
      00000000G 9F      7E D4 00271 CLRL -(SP) ;
      00000000G 9F      CF 9F 00273 PUSHAB SCRATCH ;
      00000000G 9F      03 FB 00277 CALLS #3, PROBE_ADDRESS ;
      00000000G 9F      50 E9 0027C BLBC R0, 10$ ;
      00000000G 9F      7E 7C 0027F CLRQ -(SP) ; 0552
      00000000G 9F      7E 7C 00281 CLRQ -(SP) ;
      00000000G 9F      7E D4 00283 CLRL -(SP) ;
      00000000G 9F      CF 9F 00285 PUSHAB FMT_INTERLOCK ;
      00000000G 9F      CF 9F 00289 PUSHAB PRINT_GENERAL ;
      00000000G 9F      CF 9F 0028D PUSHAB UBIMOD ;
      00000000G 9F      7E      CF 9A 00291 MOVZBL LUN, -(SP) ;
      00000000G 9F      05 DD 00296 PUSHL #5 ;
      00000000G 9F      0A FB 00298 CALLS #10, a#DS$ERRHARD ;
50      0000G CF 0003F464 8F C9 0029F 10$: BLSL3 #259172, UNIBUS_SPACE, R0 ; 0553
60      57      05 A9 002A9 BLSW3 #5, TEMP3, (R0) ; 0554
      00000000G 9F      00 FB 002AD CALLS #0, a#DS$INLOOP ; 0558
      00000000G 9F      50 E9 002B4 BLBC R0, 11$ ;
      00000000G 9F      03      50 E9 002B4 BRW 9$ ;
      00000000G 9F      03      FF51 31 002B7 BRW 9$ ;
      00000000G 9F      03      02 DD 002BA 11$: PUSHL #2 ;

```

ZZ-ECCBA-1.4
UBI TESTS_11_15
01-04

TEST_011: Direct Data Path DATIP/DATO Test
TEST_011: Direct Data Path DATIP/DATO Test

G 5

6-Sep-1985

Fiche 2 Frame G5

Sequence 264

6-Sep-1985 17:19:55

VAX-11 Bliss-32 V4.1-746

Page 18

6-Sep-1985 17:14:57

[LEMIEUX.ECCBA.RELEASE]ECCBA5.B32;5

(3)

00000000G	9F	0B	DD	002BC
00000000G	9F	02	FB	002BE
	50	00	FB	002C5
		01	D0	002CC
		04	00	002CF

PUSHL	#11
CALLS	#2, a#DS\$ENDSUB
CALLS	#0, a#DS\$BREAK
MOVL	#1, R0
RET	

; 0559
; 0561
;

; Routine Size: 720 bytes, Routine Base: TEST_011 + 0000


```
; 0562 2 $DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Direct Data Path DATOB Test');
; 0563 2
; 0564 2 !++
; 0565 2 ! TEST DESCRIPTION:
; 0566 2 !
; 0567 2 ! This tests the ability of the UBI to perform a DATOB transaction
; 0568 2 ! through the direct data path (DDP). The transactions are made from
; 0569 2 ! longword and word aligned CMI addresses. The following data patterns
; 0570 2 ! are used:
; 0571 2 !
; 0572 2 ! pattern 1 01010101
; 0573 2 ! pattern 2 10101010
; 0574 2 ! pattern 3 00110011
; 0575 2 ! pattern 4 00001111
; 0576 2 !
; 0577 2 ! Before testing starts, the Unibus Sizer routine is called to determine
; 0578 2 ! which maps are useable.
; 0579 2 !
; 0580 2 ! ASSUMPTIONS:
; 0581 2 !
; 0582 2 ! Test 1-Test 7
; 0583 2 ! --
; 0584 2
; 0585 2 LOCAL
; 0586 2 BUF_PFN,
; 0587 2 M,
; 0588 2 MAP_NUMBER,
; 0589 2 N,
; 0590 2 TEMP,
; 0591 2 TEMP0:WORD,
; 0592 2 TEMP3:WORD, ! M12
; 0593 2 UBUS_ADDR:WORD;
; 0594 2
; 0595 2 MACRO
; 0596 2 UBUS_PF = UBUS_ADDR<9,7>%; ! Defines the Unibus page frame field
; 0597 2 UBUS_BO = UBUS_ADDR<0,9>%; ! Defines the Unibus byte number field
; 0598 2
; 0599 2 MAP
; 0600 2 SCRATCH: VECTOR[5,BYTE];
; 0601 2
; 0602 2 CODECOMMENT
; 0603 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 0604 2 'a table of useable map entries. In the test which follows, the map',
; 0605 2 'selected is drawn from this table.',
; 0606 2 '':
; 0607 2 BEGIN
; 0608 3
; 0609 3 UNIBUS_SIZER();
; 0610 3 MAP_NUMBER = .FREE_MAP[0];
; 0611 3
; 0612 3 END;
; 0613 2
; 0614 2 CODECOMMENT
; 0615 2 '':
; 0616 2
```

```
; 0617 2 'Load the data patterns into the table BYTE_PATTERN for later use.',
; 0618 2 ''
; 0619 3 BEGIN
; 0620 3
; 0621 3 BYTE_PATTERN[0] = %X'55';
; 0622 3 BYTE_PATTERN[1] = %X'AA';
; 0623 3 BYTE_PATTERN[2] = %X'33';
; 0624 3 BYTE_PATTERN[3] = %X'0F';
; 0625 3
; 0626 2 END;
; 0627 2
; 0628 2 CODECOMMENT
; 0629 2 ''
; 0630 2 'Set up the UBI and UET for a DATOB, and',
; 0631 2 'clear the CMI destination location (longword aligned for M = 0, word',
; 0632 2 'aligned for M = 1) with the appropriate data pattern. Execute the DATOB',
; 0633 2 'and check that the CMI destination location contains the data pattern.',
; 0634 2 'Repeat 10 times, once for each of the five data patterns with the two',
; 0635 2 'variations of alignment.',
; 0636 2 ''
; 0637 3 BEGIN
; 0638 3
; 0639 3 INCR M FROM 0 TO 1 DO
; 0640 4 BEGIN
; 0641 4
; 0642 4 INCR N FROM 0 TO 3 DO
; 0643 5 BEGIN
; 0644 5
; 0645 5 DO ! Scope loop begins here
; 0646 6 BEGIN
; 0647 6
; 0648 6 CODECOMMENT
; 0649 6 ''
; 0650 6 'Set up the UET address register with the Unibus',
; 0651 6 'address. Bits 9 to 15 (the Unibus page frame) contain the',
; 0652 6 '7 LSBs of the UBI map number. Bits 0 to 8 contain the byte',
; 0653 6 'number field of the CMI address.',
; 0654 6 ''
; 0655 7 BEGIN
; 0656 7
; 0657 7 SCRATCH[0] = %X'FF'; ! M12
; 0658 7 SCRATCH[1] = %X'FF'; ! A12
; 0659 7 TEMPO = 0;
; 0660 7 TEMPO + .M = .BYTE_PATTERN[.N]; ! M12
; 0661 7 UET_DATA_REG = .TEMPO;
; 0662 7 UBUS_PF = .MAP_NUMBER;
; 0663 7 UBUS_BO = SCRATCH[.M];
; 0664 7 UET_ADDR_REG = .UBUS_ADDR;
; 0665 7
; 0666 6 END;
; 0667 6
; 0668 6 CODECOMMENT
; 0669 6 ''
; 0670 6 'Write the two MSBs of the Unibus address with the two',
; 0671 6 'MSBs of the map number.',
```

```

; 0672 6      ``:
; 0673 7      BEGIN
; 0674 7
; 0675 7      TEMP3 = .MAP_NUMBER<7,2> * 8;
; 0676 7
; 0677 6      END;
; 0678 6
; 0679 6      CODECOMMENT
; 0680 6      ``:
; 0681 6      'Set up the UBI map. Put the CMI page frame number (PFN)',
; 0682 6      'into the register BUF_PFN. Then call the map setup routine',
; 0683 6      'with the parameters map number, PFN of the CMI buffer, byte',
; 0684 6      'offset mode (0 = no address offset), and data path number',
; 0685 6      '(0 is DDP).',
; 0686 6      ``:
; 0687 7      BEGIN
; 0688 7
; 0689 7      TEMP = SCRATCH[.M];
; 0690 7      BUF_PFN = .TEMP<9,15>;
; 0691 7      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,0);
; 0692 7
; 0693 6      END;
; 0694 6
; 0695 6      CODECOMMENT
; 0696 6      ``:
; 0697 6      'Execute the DATOB and after a nominal wait check that the',
; 0698 6      'CMI destination contains the correct data. If not, report',
; 0699 6      'the error.',
; 0700 6      ``:
; 0701 7      BEGIN
; 0702 7
; 0703 7      UET_CTRL_REG = .TEMP3 OR DATOB;
; 0704 7
; 0705 7      $DS_WAITUS(TIME=10);
; L 0706 8      UET_STATUS(UBIMOD_DDP);          ! M7
; %PRINT:      Test 12, Subtest 0, Error 1
; 0707 7      TEMP3 = %X'FFFF';          ! A12
; 0708 7      IF .M EQL 0          ! A12
; 0709 7      THEN TEMP3<0,8> = .BYTE_PATTERN[.N] ! A12
; 0710 7      ELSE TEMP3<8,8> = .BYTE_PATTERN[.N]; ! A12
; 0711 7
; 0712 7      IF .(SCRATCH[0])<0,16> NEQ .TEMP3 ! M12
; 0713 7      THEN
; 0714 8      BEGIN
; P 0715 8      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
; P 0716 8      PRLINK = PRINT_GENERAL,
; P 0717 8      P1 = FMT_DDP_DATOB,P2 = 4,
; P 0718 8      P3 = SCRATCH[0],P4 = .M,
; P 0719 8      P5 = .TEMP3,
; L 0720 8      P6 = .(SCRATCH[0])<0,16>; ! M12
; %PRINT:      Test 12, Subtest 0, Error 2
; 0721 7      END;
; 0722 7
; 0723 6      END;
; 0724 6

```



```

: 0725 6      END
: 0726 5      WHILE $DS_INLOOP;
: 0727 5
: 0728 4      END;
: 0729 4
: 0730 3      END;
: 0731 3
: 0732 2      END;
: 0733 2
: 0734 1      $DS_ENDTEST;

```

```

                                .PSECT DISPATCH,NOWRT,NOEXE,2
                                00000000' 00000000' 00000000' 00000000' 00018 $: .ADDRESS P.AAD, P.AAE, P.AAF, TEST_012
                                00000000 00000003 00028 .LONG 3, 0
                                .PSECT $PLIT$,NOWRT,NOEXE,2
61 50 20 61 74 61 44 20 74 63 65 72 69 44 1B 000C 00028 P.AAD: .WORD 12
74 73 65 54 20 42 4F 54 41 44 20 68 74 0002A P.AAE: .ASCII <27>\Direct Data Path DATOB Test\
                                00039
                                00046 .BLKB 2
                                00048 P.AAF: .LONG 0
                                .PSECT TEST_012,NOWRT,2
                                OFFC 00000
                                5B 0000G CF 9E 00002 MOVAB UNIBUS_SPACE, R11
                                5A 0000G CF 9E 00007 MOVAB SCRATCH, R10
                                59 0000' CF 9E 0000C MOVAB BYTE_PATTERN, R9
                                5E 04 C2 00011 SUBL2 #4, SP
; Call the Unibus sizer routine to locate any Unibus memory. T
; his builds
; a table of useable map entries. In the test which follows, t
; he map
; selected is drawn from this table.
                                0000G CF 00 FB 00014 CALLS #0, UNIBUS_SIZER ; 0610
                                55 0000G CF 3C 00019 MOVZWL FREE_MAP, MAP_NUMBER ; 0611
; Load the data patterns into the table BYTE_PATTERN for later
; use.
                                04 69 00AA0055 8F D0 0001E MOVL #11141205, BYTE_PATTERN ; 0621
                                A9 000F0033 8F D0 00025 MOVL #983091, BYTE_PATTERN+4 ; 0623
; Set up the UBI and UET for a DATOB, and
; clear the CMI destination location (longword aligned for M =
; 0, word

```

```

; aligned for M = 1) with the appropriate data pattern. Execute
; the DATOB
; and check that the CMI destination location contains the data
; pattern.
; Repeat 10 times, once for each of the five data patterns with
; the two
; variations of alignment.
;
52 D4 0002D CLRL M ; 0639
53 D4 0002F 1$: CLRL N ; 0642
;
; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain the
; 7 LSBs of the UBI map number. Bits 0 to 8 contain the byte
; number field of the CMI address.
;
6A FFFF 8F B0 00031 2$: MOVW #65535, SCRATCH ; 0657
6E B4 00036 CLRW TEMP0 ; 0659
6E42 9F 00038 PUSHAB TEMP0[M] ; 0660
6943 3C 0003B MOVZWL BYTE_PATTERN[N], a(SP)+
50 6B 0003F462 8F C9 0003F BISL3 #259170, UNIBUS_SPACE, R0 ;
60 6E B0 00047 MOVW TEMP0, (R0) ; 0661
56 07 09 55 F0 0004A INSV MAP_NUMBER, #9, #7, UNBUS_ADDR ; 0662
50 52 5A C1 0004F ADDL3 R10, M, R0 ; 0663
56 09 00 F0 00053 INSV R0, #0, #9, UNBUS_ADDR ;
50 6B 0003F460 8F C9 00058 BISL3 #259168, UNIBUS_SPACE, R0 ;
60 56 B0 00060 MOVW UNBUS_ADDR, (R0) ; 0664
;
; Write the two MSBs of the Unibus address with the two
; MSBs of the map number.
;
50 55 02 07 EF 00063 EXTZV #7, #2, MAP_NUMBER, R0 ; 0675
54 50 08 A5 00068 MULW3 #8, R0, TEMP3 ;
;
; Set up the UBI map. Put the CMI page frame number (PFN)
; into the register BUF_PFN. Then call the map setup routine
; with the parameters map number, PFN of the CMI buffer, byte
; offset mode (0 = no address offset), and data path number
; (0 is DDP).
;
58 57 52 5A C1 0006C ADDL3 R10, M, TEMP ; 0689
57 0F 09 EF 00070 EXTZV #9, #15, TEMP, BUF_PFN ; 0690
7E 7C 00075 CLRQ -(SP) ; 0691
0000G CF 0120 8F BB 00077 PUSHR #^M<R5,R8> ;
04 FB 0007B CALLS #4, MAP_SETUP ;
;
; Execute the DATOB and after a nominal wait check that the
; CMI destination contains the correct data. If not, report
; the error.
;
50 6B 0003F464 8F C9 00080 BISL3 #259172, UNIBUS_SPACE, R0 ; 0701
60 54 07 A9 00088 BISW3 #7, TEMP3, (R0) ; 0703
7E 0A 7D 0008C MOVQ #10, -(SP) ; 0705
00000000G 9F 02 FB 0008F CALLS #2, a#DS$WAITUS ;
50 6B 0003F464 8F C9 00096 BISL3 #259172, UNIBUS_SPACE, R0 ; 0706

```

08	A9	60	B0	0009E	MOVW	(R0), STATUS0	:		
08	A9	8F	AA	000A2	BICW2	#65310, STATUS0	:		
	50	08	A9	32	000A8	CVTWL	STATUS0, R0	:	
			21	13	000AC	BEQL	3\$	:	
			7E	7C	000AE	CLRQ	-(SP)	:	
			50	DD	000B0	PUSHL	R0	:	
	7E		02	7D	000B2	MOVQ	#2, -(SP)	:	
		0000G	CF	9F	000B5	PUSHAB	FMT_UET_STAT	:	
		0000G	CF	9F	000B9	PUSHAB	PRINT_GENERAL	:	
		0000G	CF	9F	000BD	PUSHAB	UBIMOD_DDP	:	
	7E	0000G	CF	9A	000C1	MOVZBL	LUN, -(SP)	:	
			01	DD	000C6	PUSHL	#1	:	
00000000G	9F		0A	FB	000C8	CALLS	#10, a#DS\$ERRHARD	:	
	54		01	AE	000CF	MNEGW	#1, TEMP3	: 0707	
			52	D5	000D2	TSTL	M	: 0708	
			06	12	000D4	BNEQ	4\$	:	
	54		6943	33	000D6	CVTWB	BYTE_PATTERN[N], TEMP3	: 0709	
			08	11	000DA	BRB	5\$	:	
			6943	3F	000DC	PUSHAW	BYTE_PATTERN[N]	: 0710	
54			9E	F0	000DF	INSV	a(SP)+, #8, #8, TEMP3	:	
	08		6A	B1	000E4	CMPW	SCRATCH, TEMP3	: 0712	
	54		26	13	000E7	BEQL	6\$	:	
	7E		6A	3C	000E9	MOVZWL	SCRATCH, -(SP)	: 0720	
	7E		54	3C	000EC	MOVZWL	TEMP3, -(SP)	:	
			52	DD	000EF	PUSHL	M	:	
			5A	DD	000F1	PUSHL	R10	:	
			04	DD	000F3	PUSHL	#4	:	
		0000G	CF	9F	000F5	PUSHAB	FMT_DDP_DATOB	:	
		0000G	CF	9F	000F9	PUSHAB	PRINT_GENERAL	:	
		0000G	CF	9F	000FD	PUSHAB	UBIMOD_DDP	:	
	7E	0000G	CF	9A	00101	MOVZBL	LUN, -(SP)	:	
			02	DD	00106	PUSHL	#2	:	
00000000G	9F		0A	FB	00108	CALLS	#10, a#DS\$ERRHARD	:	
00000000G	9F		00	FB	0010F	CALLS	#0, a#DS\$INLOOP	: 0726	
	03		50	E9	00116	BLBC	R0, 8\$	:	
			FF15	31	00119	BRW	2\$	:	
	53		03	F3	0011C	AOBLEQ	#3, N, 7\$	: 0642	
FF09	F9		01	F1	00120	ACBL	#1, #1, M, 1\$	: 0639	
	52		00	FB	00126	CALLS	#0, a#DS\$BREAK	: 0732	
		00000000G	9F	00	FB	00126	CALLS	#0, a#DS\$BREAK	: 0732
			50	01	D0	0012D	MOVL	#1, R0	: 0734
			04	00	00130	RET		:	

; Routine Size: 305 bytes, Routine Base: TEST_012 + 0000

\$DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Buffered Address Register Test');

!++
! TEST DESCRIPTION:

This tests the integrity of the buffered address registers (BARs). These three registers, one per BDP, are located in the UDP chips. They are bit sliced, with four bits per UDP. For diagnostic reference, the UDPs are assigned the logical designations UDP0, UDP1, UDP2, and UDP3. The Unibus address bits corresponding to each chip are as follows:

Unibus address bits in UDP					
UDP#	BAR3	BAR2	BAR1	BAR0	BAR bit
0	16	15	03	02	
1	08	17	10	09	
2	05	04	12	11	
3	07	06	14	13	

There are two types of faults that can occur here. In the first type the BAR shows an address match when there should not be one, and in the second type of error the BAR fails to indicate a match when there should be one. The first two subtests deal with the first variety of fault, and the second two subtests deal with the second variety of fault.

The first subtest deals with the case of the address match occurring when there should not be a match due to a stuck or shorted bit in the byte number field (bits 2 through 8) of the BAR. In order to detect this condition, a sequence of unique data patterns is written into a page of memory at following sequence of byte number addresses:

CMI byte number address (binary)	data pattern
000000000	0
000000100	2
000001000	3
000010000	4
000100000	5
001000000	6
010000000	7
100000000	8

A DATI is then executed to read each of these patterns, one at a time, through each of the three BDPs. The contents of the UET data register are then verified. This address sequence causes a one to be floated across the byte number field of the BAR.

```

; 0790 2  !
; 0791 2  !
; 0792 2  ! The second subtest causes a one to be floated across the page frame
; 0793 2  ! field of the BAR by executing DATIs with each map of the set 0, 1, 2,
; 0794 2  ! 4, 8, 16, 32, 64, 128, and 256.
; 0795 2  !
; 0796 2  ! The third and fourth subtests deal with the case of the BAR failing to
; 0797 2  ! indicate an address match when it should. In order to detect this
; 0798 2  ! type of fault, it is necessary to execute a DATI from a particular CMI
; 0799 2  ! location, change the data in that CMI location, and execute the DATI
; 0800 2  ! again without changing the address. If the BAR properly indicates an
; 0801 2  ! address match, the data in the UET data register would not
; 0802 2  ! change after the first DATI. However, if the BAR contained a stuck or
; 0803 2  ! shorted bit which caused a failure to indicate a match in this case,
; 0804 2  ! the data in the UET data register would change after the second
; 0805 2  ! DATI, revealing the fault. In order to check each bit of the BAR,
; 0806 2  ! this procedure is followed for the same addressing patterns as in the
; 0807 2  ! preceding two subtests.
; 0808 2  !
; 0809 2  ! Before testing starts, the Unibus Sizer routine is called to build a
; 0810 2  ! table of useable maps. This test determines which map entries can be
; 0811 2  ! used in the test based on the contents of the table.
; 0812 2  ! ASSUMPTIONS:
; 0813 2  ! --
; 0814 2  !
; 0815 2  REGISTER
; 0816 2  BUF_PFN,
; 0817 2  I,
; 0818 2  MAP_ENTRY,
; 0819 2  MAP_NUMBER,
; 0820 2  M:WORD,
; 0821 2  N,
; 0822 2  UBUS_ADDR: WORD,
; 0823 2  TEMP,
; 0824 2  TEMP1:WORD,
; 0825 2  TEMP2:WORD,
; 0826 2  TEMP3:WORD;
; 0827 2  !
; 0828 2  MACRO
; 0829 2  V_BIT = MAP_ENTRY<31,1>%, ! Defines the UBI map valid bit field ! M7
; 0830 2  UBUS_PF = UBUS_ADDR<9,7>%, ! Defines the Unibus page frame field
; 0831 2  UBUS_BO = UBUS_ADDR<0,9>%; ! Defines the Unibus byte number field
; 0832 2  !
; 0833 2  CODECOMMENT
; 0834 2  'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 0835 2  'a bit map of useable map entries. In the test which follows, the maps',
; 0836 2  'to be used are checked against this bitvector before they are used.',
; 0837 2  '
; 0838 2  '
; 0839 3  BEGIN
; 0840 3  UNIBUS_SIZER();
; 0841 3  UNIBUS_SIZER();
; 0842 3  UNIBUS_SIZER();
; 0843 2  END;
; 0844 2

```



```

: 0845 3 $DS_BGNSUB; ! Subtest 1
: 0846 3
: 0847 3 CODECOMMENT
: 0848 3 ''
: 0849 3 'Initialize the UET data register and store the data patterns to',
: 0850 3 'be used into SCRATCH.',
: 0851 3 ''
: 0852 4 BEGIN
: 0853 4
: 0854 4 UET_DATA_REG = -1;
: 0855 4 SCRATCH[0] = 0; ! Patterns to cause a floating one
: 0856 4 SCRATCH[1] = 2; ! in the byte number address
: 0857 4 SCRATCH[2] = 3;
: 0858 4 SCRATCH[4] = 4;
: 0859 4 SCRATCH[8] = 5;
: 0860 4 SCRATCH[16] = 6;
: 0861 4 SCRATCH[32] = 7;
: 0862 4 SCRATCH[64] = 8;
: 0863 4
: 0864 3 END;
: 0865 3
: 0866 3 CODECOMMENT
: 0867 3 ''
: 0868 3 'Float a one through each byte number bit of the bits 2 through 8 of',
: 0869 3 'each BAR. Set up a DAT1 and initiate it for each of the longword',
: 0870 3 'elements of the vector SCRATCH in the sequence 0, 1, 2, 4, 8, 16, 32,',
: 0871 3 '64. After the DAT1, check the UET data register for the correct',
: 0872 3 'contents. A stuck or shorted byte number address bit will show up as',
: 0873 3 'incorrect data in the UET data register. Repeat this sequence',
: 0874 3 'for each BDP.',
: 0875 3 ''
: 0876 4 BEGIN
: 0877 4
: 0878 4 MAP_NUMBER = .FREE_MAP[0];
: 0879 4 INCR I FROM 1 TO 3 DO
: 0880 5 BEGIN
: 0881 5
: 0882 5 INCR M FROM 0 TO 7 DO
: 0883 6 BEGIN
: 0884 6
: 0885 6 DO ! Scope loop starts here
: 0886 7 BEGIN
: 0887 7 UBI_CSR(.I) = PURGE; ! Purge the data path A7
: 0888 7 CODECOMMENT
: 0889 7 ''
: 0890 7 'Set up the UET address register with the Unibus',
: 0891 7 'address. Bits 9 to 15 (the Unibus page frame) contain the',
: 0892 7 '7 LSBs of the UBI map number. Bits 0 to 8 contain the byte',
: 0893 7 'number field of the CMI address.',
: 0894 7 ''
: 0895 8 BEGIN
: 0896 8
: 0897 8 UBUS_PF = .MAP_NUMBER;
: 0898 8 N = 1 ^ (.M - 1);
: 0899 8 UBUS_BO = SCRATCH[.N];

```



```
; 0900 8          UET_ADDR_REG = .UBUS_ADDR;
; 0901 8
; 0902 7          END;
; 0903 7
; 0904 7          CODECOMMENT
; 0905 7          'Write the two MSBs of the Unibus address with the two MSBs',
; 0906 7          'of the map number.',
; 0907 7          '
; 0908 7          '
; 0909 8          BEGIN
; 0910 8
; 0911 8          TEMP3 = .MAP_NUMBER<7,2> * 8;
; 0912 8
; 0913 7          END;
; 0914 7
; 0915 7          CODECOMMENT
; 0916 7          'Set up the UBI map. Put the CMI page frame number (PFN)',
; 0917 7          'into the register BUF_PFN. Then call the map setup routine',
; 0918 7          'with the parameters map number, PFN of the CMI buffer, byte',
; 0919 7          'offset mode (0 = no address offset), and data path number.',
; 0920 7          '
; 0921 7          '
; 0922 8          BEGIN
; 0923 8
; 0924 8          TEMP = SCRATCH;
; 0925 8          BUF_PFN = .TEMP<9,15>;
; 0926 8          MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.I);
; 0927 8
; 0928 7          END;
; 0929 7
; 0930 7          CODECOMMENT
; 0931 7          'Execute a DATI and after a nominal wait compare the data in',
; 0932 7          'the UET data register with the expected data. If',
; 0933 7          'there is an error, report it.',
; 0934 7          '
; 0935 7          '
; 0936 8          BEGIN
; 0937 8
; 0938 8          UET_CTRL_REG = .TEMP3 OR DATI;
; 0939 8
; 0940 8          $DS_WAITUS(TIME=10);
; L 0941 8          UBI_STATUS(.I,UBIMOD_BYTE_OFF);          ! M7
; %PRINT:          Test 13, Subtest 1, Error 1
; L 0942 9          UET_STATUS(UBIMOD_BYTE_OFF);          ! M7
; %PRINT:          Test 13, Subtest 1, Error 2
; 0943 8          TEMP1 = .UET_DATA_REG;
; 0944 8          TEMP2 = .SCRATCH[N];
; 0945 8          IF .TEMP1 NEQ .TEMP2
; 0946 8          THEN
; 0947 9          BEGIN
; P 0948 9          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BYTE_OFF,
; P 0949 9          PRLINK = PRINT_GENERAL,
; P 0950 9          P1 = FMT_BYTE_OFF,P2 = 2,
; L 0951 9          P3 = .TEMP2,P4 = .TEMP1);
; %PRINT:          Test 13, Subtest 1, Error 3
```

```

: 0952 8          END;
: 0953 8
: 0954 7          END;
: 0955 7
: 0956 7          END
: 0957 6      WHILE $DS_INLOOP;    ! Scope loop ends here
: 0958 6
: 0959 5          END;
: 0960 5
: 0961 4          END;
: 0962 4
: 0963 3          END;
: 0964 2      $DS_ENDSUB;
: 0965 2
: 0966 3      $DS_BGNSUB;          ! Subtest 2
: 0967 3
: 0968 3      CODECOMMENT
: 0969 3      ``
: 0970 3      'Invalidate all map entries and initialize the UET data register.',
: 0971 3      ``
: 0972 3
: 0973 4          BEGIN
: 0974 4      UET_DATA_REG = -1;
: 0975 4      V_BIT = 0;
: 0976 4      INCR N FROM 0 TO 511 DO
: 0977 5          BEGIN
: 0978 5      UBI_MAP(.N) = .MAP_ENTRY;
: 0979 4          END;
: 0980 3      END;
: 0981 3
: 0982 3      CODECOMMENT
: 0983 3      ``
: 0984 3      'Write a unique data pattern to a memory location, set up and execute a',
: 0985 3      'DATI from that location, and check that the UET data register',
: 0986 3      'contains the correct data. If it does not, then the correct map has',
: 0987 3      'not been addressed properly due to a stuck or shorted Unibus page frame',
: 0988 3      'bit. This sequence is repeated for each map entry of the set of map',
: 0989 3      'entries 0, 1, 2, 4, 8, 16, 32, 64, 128, and 256.',
: 0990 3      ``
: 0991 4          BEGIN
: 0992 4
: 0993 4      INCR I FROM 1 TO 3 DO
: 0994 5          BEGIN
: 0995 5
: 0996 5      INCR M FROM 0 TO 9 DO
: 0997 6          BEGIN
: 0998 6
: 0999 6      N = 1 ^ (.M - 1);
: 1000 6      IF .VALID_MAPS[.N]
: 1001 6      THEN
: 1002 7          BEGIN
: 1003 7
: 1004 7      DO          ! Scope loop starts here
: 1005 8          BEGIN
: 1006 8      UBI_CSR(.I) = PURGE;    ! Purge the data path

```

: 1007 8
: 1008 8
: 1009 8
: 1010 8
: 1011 8
: 1012 8
: 1013 8
: 1014 8
: 1015 9
: 1016 9
: 1017 9
: 1018 9
: 1019 9
: 1020 9
: 1021 8
: 1022 8
: 1023 8
: 1024 8
: 1025 8
: 1026 8
: 1027 8
: 1028 9
: 1029 9
: 1030 9
: 1031 9
: 1032 8
: 1033 8
: 1034 8
: 1035 8
: 1036 8
: 1037 8
: 1038 8
: 1039 8
: 1040 8
: 1041 8
: 1042 9
: 1043 9
: 1044 9
: 1045 9
: 1046 9
: 1047 9
: 1048 9
: 1049 8
: 1050 8
: 1051 8
: 1052 8
: 1053 8
: 1054 8
: 1055 8
: 1056 8
: 1057 9
: 1058 9
: 1059 9
: 1060 9
: 1061 9

CODECOMMENT

Set up the UET address register with the Unibus',
address. Bits 9 to 15 (the Unibus page frame) contain',
the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
the byte number field of the CMI address.'

BEGIN

UBUS_PF = .N;
UBUS_BO = SCRATCH;
UET_ADDR_REG = .UBUS_ADDR;

END;

CODECOMMENT

Write the two MSBs of the Unibus address with the two',
MSBs of the map number.'

BEGIN

TEMP3 = .N<7,2> * 8;

END;

CODECOMMENT

Set up the UBI map. Put the CMI page frame number',
(PFN) into the register BUF_PFN. Then call the map',
setup routine with the parameters map number, PFN of',
the CMI buffer, byte offset mode (0 = no address',
offset), and data path number.'

BEGIN

TEMP = SCRATCH;
BUF_PFN = .TEMP<9,15>;
MAP_SETUP(.N,.BUF_PFN,0,.I);
SCRATCH = .M;

END;

CODECOMMENT

Execute a DATI and after a nominal wait compare the',
data in the UET data register with the expected',
data. If there is an error, report it.'

BEGIN

UET_CTRL_REG = .TEMP3 OR DATI;

\$DS_WAITUS(TIME=10);


```
; L 1062 9          UBI_STATUS(.I,UBIMOD_UB_CMI);      ! M7
; %PRINT:          Test 13, Subtest 2, Error 4
; L 1063 10         UET_STATUS(UBIMOD_UB_CMI);          ! M7
; %PRINT:          Test 13, Subtest 2, Error 5
; 1064 9           TEMP1 = .UET_DATA_REG;
; 1065 9           IF .TEMP1 NEQ .M
; 1066 9           THEN
; 1067 10           BEGIN
; P 1068 10           $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_UB_CMI,
; P 1069 10           PRLINK = PRINT_GENERAL,
; P 1070 10           P1 = FMT_UB_CMI,P2 = 2,
; L 1071 10           P3 = .M,P4 = .TEMP1);
; %PRINT:          Test 13, Subtest 2, Error 6
; 1072 9           END;
; 1073 9           UBI_MAP(.N) = .MAP_ENTRY;            ! Invalidate the map
; 1074 9
; 1075 8           END;
; 1076 8
; 1077 8           END
; 1078 7           WHILE $DS_INLOOP;                    ! Scope loop ends here
; 1079 7
; 1080 6           END;
; 1081 6
; 1082 5           END;
; 1083 5
; 1084 4           END;
; 1085 4
; 1086 3           END;
; 1087 2           $DS_ENDSUB;
; 1088 2
; 1089 3           $DS_BGNSUB;                          ! Subtest 3
; 1090 3
; 1091 3           CODECOMMENT
; 1092 3           ''
; 1093 3           'Float a one through each byte number bit of the bits 2 through 8 of',
; 1094 3           'each BAR. Set up a DATI and initiate it for each of the longword',
; 1095 3           'elements of the vector SCRATCH in the sequence 0, 1, 2, 4, 8, 16, 32,',
; 1096 3           '64. After the DATI, change the contents of the CMI location, execute',
; 1097 3           'another DATI, and check that the contents of the UET data',
; 1098 3           'register did not change. If it did, there is a stuck or shorted BAR',
; 1099 3           'bit. Repeat this sequence for each BDP.',
; 1100 3           ''
; 1101 4           BEGIN
; 1102 4
; 1103 4           INCR I FROM 1 TO 3 DO
; 1104 5           BEGIN
; 1105 5
; 1106 5           INCR M FROM 0 TO 7 DO
; 1107 6           BEGIN
; 1108 6
; 1109 6           DO                                    ! Scope loop starts here
; 1110 7           BEGIN
; 1111 7           UBI_CSR(.I) = PURGE;                  ! Purge the data path
; 1112 7
; 1113 7           CODECOMMENT
```

A7

```

; 1114 7
; 1115 7
; 1116 7
; 1117 7
; 1118 7
; 1119 7
; 1120 8
; 1121 8
; 1122 8
; 1123 8
; 1124 8
; 1125 8
; 1126 8
; 1127 7
; 1128 7
; 1129 7
; 1130 7
; 1131 7
; 1132 7
; 1133 7
; 1134 8
; 1135 8
; 1136 8
; 1137 8
; 1138 7
; 1139 7
; 1140 7
; 1141 7
; 1142 7
; 1143 7
; 1144 7
; 1145 7
; 1146 7
; 1147 7
; 1148 8
; 1149 8
; 1150 8
; 1151 8
; 1152 8
; 1153 8
; 1154 8
; 1155 7
; 1156 7
; 1157 7
; 1158 7
; 1159 7
; 1160 7
; 1161 7
; 1162 7
; 1163 7
; 1164 8
; 1165 8
; 1166 8
; 1167 8
; 1168 8

''
'Set up the UET address register with the Unibus',
'address. Bits 9 to 15 (the Unibus page frame) contain',
'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
'the byte number field of the CMI address.',
''
BEGIN
    UBUS_PF = .MAP_NUMBER;
    N = 1 ^ (.M - 1);
    UBUS_BO = SCRATCH[N];
    UET_ADDR_REG = .UBUS_ADDR;
END;

CODECOMMENT
''
'Write the two MSBs of the Unibus address with the two',
'MSBs of the map number.',
''
BEGIN
    TEMP3 = .MAP_NUMBER<7,2> * 8;
END;

CODECOMMENT
''
'Set up the UBI map. Put the CMI page frame number',
'(PFN) into the register BUF_PFN. Then call the map',
'setup routine with the parameters map number, PFN of',
'the CMI buffer, byte offset mode (0 = no address',
'offset), and data path number.',
''
BEGIN
    TEMP = SCRATCH;
    BUF_PFN = .TEMP<9,15>;
    MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.1);
    SCRATCH[N] = .M;
END;

CODECOMMENT
''
'Execute a DATI and after a nominal wait, change the data in',
'the CMI location and execute a second DATI. Then check',
'the data in the UET data register did not change.',
'If it did, report the error.',
''
BEGIN
    UET_CTRL_REG = .TEMP3 OR DATI;
    $DS_WAITUS(TIME=10);

```

```

; L 1169 8      UBI_STATUS(.I,UBIMOD_BYTE_OFF);      ! M7
; xPRINT:      Test 13, Subtest 3, Error 7
; L 1170 9      UET_STATUS(UBIMOD_BYTE_OFF);          ! M7
; xPRINT:      Test 13, Subtest 3, Error 8
; L 1171 8      SCRATCH(.N) = NOT .M;
; L 1172 8      UET_ADDR_REG = .UBUS_ADDR;
; L 1173 8
; L 1174 8      UET_CTRL_REG = .TEMP3 OR DAT1;
; L 1175 8
; L 1176 8      $DS_WAITUS(TIME=10);
; L 1177 8      UBI_STATUS(.I,UBIMOD_BYTE_OFF);          ! M7
; xPRINT:      Test 13, Subtest 3, Error 9
; L 1178 9      UET_STATUS(UBIMOD_BYTE_OFF);          ! M7
; xPRINT:      Test 13, Subtest 3, Error 10
; L 1179 8      TEMP1 = .UET_DATA_REG;
; L 1180 8      TEMP2 = .SCRATCH(.N);
; L 1181 8      IF .TEMP1 EQL .TEMP2
; L 1182 8      THEN
; L 1183 9      BEGIN
; P 1184 9      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BYTE_OFF,
; P 1185 9      PRLINK = PRINT_GENERAL,
; P 1186 9      P1 = FMT_BYTE_OFF,P2 = 2,
; L 1187 9      P3 = .TEMP2,P4 = .TEMP1);
; xPRINT:      Test 13, Subtest 3, Error 11
; L 1188 8      END;
; L 1189 8
; L 1190 7      END;
; L 1191 7
; L 1192 7      END
; L 1193 6      WHILE $DS_INLOOP;      ! Scope loop ends here
; L 1194 6
; L 1195 5      END;
; L 1196 5
; L 1197 4      END;
; L 1198 4
; L 1199 3      END;
; L 1200 2      $DS_ENDSUB;
; L 1201 2
; L 1202 3      $DS_BGNSUB;      ! Subtest 4
; L 1203 3
; L 1204 3      CODECOMMENT
; L 1205 3      'Invalidate all map entries and initialize the UET data register.'
; L 1206 3      '
; L 1207 3      '
; L 1208 4      BEGIN
; L 1209 4
; L 1210 4      UET_DATA_REG = -1;
; L 1211 4      V_BIT = 0;
; L 1212 4      INCR N FROM 0 TO 511 DO
; L 1213 5      BEGIN
; L 1214 5
; L 1215 5      UBI_MAP(.N) = .MAP_ENTRY;
; L 1216 5
; L 1217 4      END;
; L 1218 4

```



```

: 1219 3      END;
: 1220 3
: 1221 3      CODECOMMENT
: 1222 3      ''
: 1223 3      'Write a unique data pattern to a memory location, set up and execute a',
: 1224 3      'DATI from that location, and check that the UET data register',
: 1225 3      'contains the correct data. If it does not, then the correct map has',
: 1226 3      'not been addressed properly due to a stuck or shorted Unibus page frame',
: 1227 3      'bit. This sequence is repeated for each map entry of the set of map',
: 1228 3      'entries 0, 1, 2, 4, 8, 16, 32, 64, 128, and 256.',
: 1229 3      ''
: 1230 4      BEGIN
: 1231 4
: 1232 4      INCR I FROM 1 TO 3 DO
: 1233 5          BEGIN
: 1234 5
: 1235 5          INCR M FROM 0 TO 9 DO
: 1236 6              BEGIN
: 1237 6
: 1238 6              N = 1 ^ (.M - 1);
: 1239 6              IF .VALID_MAPS[N]
: 1240 6              THEN
: 1241 7                  BEGIN
: 1242 7
: 1243 7                      DO                      ! Scope loop begins here
: 1244 8                          BEGIN
: 1245 8                          UBI_CSR(.I) = PURGE;          ! Purge the data path          A7
: 1246 8
: 1247 8                          CODECOMMENT
: 1248 8                          ''
: 1249 8                          'Set up the UET address register with the Unibus',
: 1250 8                          'address. Bits 9 to 15 (the Unibus page frame) contain',
: 1251 8                          'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
: 1252 8                          'the byte number field of the CMI address.',
: 1253 8                          ''
: 1254 9                          BEGIN
: 1255 9
: 1256 9                          UBUS_PF = .N;
: 1257 9                          UBUS_BO = SCRATCH;
: 1258 9                          UET_ADDR_REG = .UBUS_ADDR;
: 1259 9
: 1260 8                          END;
: 1261 8
: 1262 8                          CODECOMMENT
: 1263 8                          ''
: 1264 8                          'Write the two MSBs of the Unibus address with the two',
: 1265 8                          'MSBs of the map number.',
: 1266 8                          ''
: 1267 9                          BEGIN
: 1268 9
: 1269 9                          TEMP3 = .N<7,2> * 8;
: 1270 9
: 1271 8                          END;
: 1272 8
: 1273 8                          CODECOMMENT

```

```

; 1274 8
; 1275 8
; 1276 8
; 1277 8
; 1278 8
; 1279 8
; 1280 8
; 1281 9
; 1282 9
; 1283 9
; 1284 9
; 1285 9
; 1286 9
; 1287 9
; 1288 8
; 1289 8
; 1290 8
; 1291 8
; 1292 8
; 1293 8
; 1294 8
; 1295 8
; 1296 8
; 1297 9
; 1298 9
; 1299 9
; 1300 9
; 1301 9
; L 1302 9
; xPRINT:
; L 1303 10
; xPRINT:
; 1304 9
; 1305 9
; 1306 9
; 1307 9
; 1308 9
; 1309 9
; L 1310 9
; xPRINT:
; L 1311 10
; xPRINT:
; 1312 9
; 1313 9
; 1314 9
; 1315 10
; P 1316 10
; P 1317 10
; P 1318 10
; L 1319 10
; xPRINT:
; 1320 9
; 1321 9
; 1322 9
; 1323 8

      ``
      'Set up the UBI map. Put the CMI page frame number',
      '(PFN) into the register BUF_PFN. Then call the map',
      'setup routine with the parameters map number, PFN of',
      'the CMI buffer, byte offset mode (0 = no address',
      'offset), and data path number.',
      ``
      BEGIN

      TEMP = SCRATCH;
      BUF_PFN = .TEMP<9,15>;
      MAP_SETUP(.N,.BUF_PFN,0,.I);
      SCRATCH = .M;

      END;

CODECOMMENT
      ``
      'Execute a DATI and after a nominal wait, change the',
      'data in the CMI location and execute a second DATI.',
      'Then check the data in the UET data register',
      'did not change. If it did, report the error.',
      ``
      BEGIN

      UET_CTRL_REG = .TEMP3 OR DATI;

      $DS_WAITUS(TIME=10);
      UBI_STATUS(.I,UBIMOD_UB_CMI);      ! M7
      UET_STATUS(UBIMOD_UB_CMI);      ! M7
      SCRATCH = NOT .M;
      UET_ADDR_REG = .UBUS_ADDR;

      UET_CTRL_REG = .TEMP3 OR DATI;

      $DS_WAITUS(TIME=10);
      UBI_STATUS(.I,UBIMOD_UB_CMI);      ! M7
      UET_STATUS(UBIMOD_UB_CMI);      ! M7
      TEMP1 = .UET_DATA_REG;
      IF .TEMP1 NEQ .M
      THEN
      BEGIN
      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_UB_CMI,
      PRLINK = PRINT_GENERAL,
      P1 = FMT_UB_CMI,P2 = 2,
      P3 = .M,P4 = .TEMP1);
      END;
      UBI_MAP(.N) = .MAP_ENTRY;      ! Invalidate the map

      END;

```

Test 13, Subtest 4, Error 12

Test 13, Subtest 4, Error 13

Test 13, Subtest 4, Error 14

Test 13, Subtest 4, Error 15

Test 13, Subtest 4, Error 16

```
; 1324 8
; 1325 8
; 1326 7
; 1327 7
; 1328 6
; 1329 6
; 1330 5
; 1331 5
; 1332 4
; 1333 4
; 1334 3
; 1335 2
; 1336 2
; 1337 1

                END
            WHILE $DS_INLOOP;        ! Scope loop ends here
        END;
    END;
END;
$DS_ENDSUB;
$DS_ENDTEST;
```

```
.PSECT DISPATCH,NOWRT,NOEXE,2
00000000' 00000000' 00000000' 00000000' 00030 $: .ADDRESS P.AAG, P.AAH, P.AAI, TEST_013
00000000 00000003 00040 .LONG 3, 0
.PSECT $PLIT$,NOWRT,NOEXE,2
000D 0004C P.AAG: .WORD 13
65 72 64 64 41 20 64 65 72 65 66 66 75 42 1E 0004E P.AAH: .ASCII <30>\Buffered Address Register Test\
73 65 54 20 72 65 74 73 69 67 65 52 20 73 73 0005D
74 0006C
00000000 00070 P.AAI: .BLKB 3
.ENTRY TEST_013,NOWRT,2
OFFC 00000 .ENTRY TEST_013, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0735
SE 0C C2 00002 SUBL2 #12, SP
; Call the Unibus sizer routine to locate any Unibus memory. T
; his builds
; a bit map of useable map entries. In the test which follows,
; the maps
; to be used are checked against this bitvector before they are
; used.
0000G CF 00 FB 00005 CALLS #0, UNIBUS_SIZER ; 0841
01 DD 0000A PUSHL #1 ; 0843
0D DD 0000C PUSHL #13
00000000G 9F 02 FB 0000E CALLS #2, a#$DS$BGNSUB
; Initialize the UET data register and store the data patterns
; to
; be used into SCRATCH.
```



```

50      0000G  CF 0003F462  8F  C9 00015
           60              01  AE 0001F
           0000G      0000G  CF  D4 00022
           0000G  CF      02  D0 00026
           0000G  CF      03  D0 0002B
           0000G  CF      04  D0 00030
           0000G  CF      05  D0 00035
           0000G  CF      06  D0 0003A
           0000G  CF      07  D0 0003F
           0000G  CF      08  D0 00044

```

```

BISL3  #259170, UNIBUS_SPACE, R0 ; 0852
MNEGW  #1, (R0) ; 0854
CLRL   SCRATCH ; 0855
MOVL   #2, SCRATCH+4 ; 0856
MOVL   #3, SCRATCH+8 ; 0857
MOVL   #4, SCRATCH+16 ; 0858
MOVL   #5, SCRATCH+32 ; 0859
MOVL   #6, SCRATCH+64 ; 0860
MOVL   #7, SCRATCH+128 ; 0861
MOVL   #8, SCRATCH+256 ; 0862

```

;
;
;
;
;
;
;

```

; Float a one through each byte number bit of the bits 2 through
; 8 of
; each BAR. Set up a DATI and initiate it for each of the long
; word
; elements of the vector SCRATCH in the sequence 0, 1, 2, 4, 8,
; 16, 32,
; 64. After the DATI, check the UET data register for the correct
; contents. A stuck or shorted byte number address bit will show up as
; incorrect data in the UET data register. Repeat this sequence
; for each BDP.

```

```

           54      0000G  CF  3C 00049
           53              01  D0 0004E
           5B      0000G  CF  D4 00051
08  AE      53              02  78 00053
50      0000G  CF      08  AE  C1 00058
           60              01  D0 0005F

```

```

MOVZWL  FREE_MAP, MAP_NUMBER ; 0878
MOVL    #1, I ; 0879
CLRL    M ; 0882
ASHL    #2, I, 8(SP) ; 0887
ADDL3   8(SP), UBI_UNDER_TEST, R0 ;
MOVL    #1, (R0) ;

```

```

; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain the
; 7 LSBs of the UBI map number. Bits 0 to 8 contain the byte
; number field of the CMI address.

```

```

56      07      09      54  F0 00062
           50      FF  AB  9E 00067
55      01      50  78 0006B
           50      0000GCF45  DE 0006F
56      09      00      50  F0 00075
           50      0000G  CF 0003F460  8F  C9 0007A
           60              56  B0 00084

```

```

INSV    MAP_NUMBER, #9, #7, UBUS_ADDR ; 0897
MOVAB   -1(R11), R0 ; 0898
ASHL    R0, #1, N ;
MOVAL   SCRATCH[N], R0 ; 0899
INSV    R0, #0, #9, UBUS_ADDR ;
BISL3   #259168, UNIBUS_SPACE, R0 ;
MOVW    UBUS_ADDR, (R0) ; 0900

```

```

; Write the two MSBs of the Unibus address with the two MSBs
; of the map number.

```

```

50      54      02      07  EF 00087
           5A      50      08  A5 0008C

```

```

EXTZV   #7, #2, MAP_NUMBER, R0 ; 0911
MULW3   #8, R0, TEMP3 ;

```

```

; Set up the UBI map. Put the CMI page frame number (PFN)
; into the register BUF_PFN. Then call the map setup routine
; with the parameters map number, PFN of the CMI buffer, byte
; offset mode (0 = no address offset), and data path number.

```

52		57	0000G	CF	9E	00090	MOVAB	SCRATCH, TEMP	: 0924		
	57	OF		09	EF	00095	EXTZV	#9, #15, TEMP, BUF_PFN	: 0925		
				53	DD	0009A	PUSHL	I	: 0926		
				7E	D4	0009C	CLRL	-(SP)	:		
				52	DD	0009E	PUSHL	BUF_PFN	:		
				54	DD	000A0	PUSHL	MAP_NUMBER	:		
		0000G	CF	04	FB	000A2	CALLS	#4, MAP_SETUP	:		
									:		
								; Execute a DATI and after a nominal wait compare the data in			
								; the UET data register with the expected data. If			
								; there is an error, report it.			
								:			
	50	0000G	CF	0003F464	8F	C9	000A7	BISL3	#259172, UNIBUS_SPACE, R0	: 0936	
	60		5A		01	A9	000B1	BISW3	#1, TEMP3, (R0)	: 0938	
			7E		0A	7D	000B5	MOVQ	#10, -(SP)	: 0940	
		00000000G	9F		02	FB	000B8	CALLS	#2, a#DS\$WAITUS	:	
08	AE		53		02	78	000BF	ASHL	#2, I, 8(SP)	: 0941	
	50	0000G	CF	08	AE	C1	000C4	ADDL3	8(SP), UBI_UNDER_TEST, R0	:	
0000'	CF		60	1FFFFFFE	8F	CB	000CB	BICL3	#536870910, (R0), STATUS1	:	
			50	0000'	CF	D0	000D5	MOVL	STATUS1, R0	:	
					20	18	000DA	BGEQ	4\$	:	
					7E	7C	000DC	CLRQ	-(SP)	:	
					7E	D4	000DE	CLRL	-(SP)	:	
					50	DD	000E0	PUSHL	R0	:	
					7E	D4	000E2	CLRL	-(SP)	:	
					53	DD	000E4	PUSHL	I	:	
				0000G	CF	9F	000E6	PUSHAB	PRINT_CSR_ERR	:	
				0000G	CF	9F	000EA	PUSHAB	UBIMOD_BYTE_OFF	:	
			7E	0000G	CF	9A	000EE	MOVZBL	LUN, -(SP)	:	
					01	DD	000F3	PUSHL	#1	:	
		00000000G	9F		0A	FB	000F5	CALLS	#10, a#DS\$ERRHARD	:	
50		0000G	CF	0003F464	8F	C9	000FC	BISL3	#259172, UNIBUS_SPACE, R0	: 0942	
		0000'	CF		60	B0	00106	MOVW	(R0), STATUS0	:	
		0000'	CF	FF1E	8F	AA	0010B	BICW2	#65310, STATUS0	:	
			50	0000'	CF	32	00112	CVTLW	STATUS0, R0	:	
					21	13	00117	BEQL	5\$	:	
					7E	7C	00119	CLRQ	-(SP)	:	
					50	DD	0011B	PUSHL	R0	:	
			7E		02	7D	0011D	MOVQ	#2, -(SP)	:	
				0000G	CF	9F	00120	PUSHAB	FMT_UET_STAT	:	
				0000G	CF	9F	00124	PUSHAB	PRINT_GENERAL	:	
				0000G	CF	9F	00128	PUSHAB	UBIMOD_BYTE_OFF	:	
			7E	0000G	CF	9A	0012C	MOVZBL	LUN, -(SP)	:	
					02	DD	00131	PUSHL	#2	:	
		00000000G	9F		0A	FB	00133	CALLS	#10, a#DS\$ERRHARD	:	
50		0000G	CF	0003F462	8F	C9	0013A	BISL3	#259170, UNIBUS_SPACE, R0	: 0943	
			58		60	B0	00144	MOVW	(R0), TEMP1	:	
			59	0000G	CF	45	F7	00147	CVTLW	SCRATCH[N], TEMP2	: 0944
			59		58	B1	0014D	CMPW	TEMP1, TEMP2	: 0945	
					24	13	00150	BEQL	6\$	:	
					7E	7C	00152	CLRQ	-(SP)	: 0951	
			7E		58	3C	00154	MOVZWL	TEMP1, -(SP)	:	
			7E		59	3C	00157	MOVZWL	TEMP2, -(SP)	:	
					02	DD	0015A	PUSHL	#2	:	


```

                                0000G CF 9F 0015C      PUSHAB FMT_BYTE_OFF      ;
                                0000G CF 9F 00160      PUSHAB PRINT_GENERAL      ;
                                0000G CF 9F 00164      PUSHAB UBIMOD_BYTE_OFF      ;
                                0000G CF 9A 00168      MOVZBL LUN, -(SP)      ;
                                03 DD 0016D      PUSHL #3      ;
                                00000000G 9F 0A FB 0016F      CALLS #10, a#DS$ERRHARD      ;
                                00000000G 9F 00 FB 00176 6$: CALLS #0, a#DS$INLOOP      ; 0957
                                03 50 E9 0017D      BLBC R0, 7$      ;
                                FECA 5B 01 FED5 31 00180      BRW 3$      ;
                                FEC2 53 01 07 F1 00183 7$: ACBL #7, #1, M, 2$      ; 0882
                                03 F1 00189      ACBL #3, #1, I, 1$      ; 0879
                                01 DD 0018F      PUSHL #1      ; 0963
                                0D DD 00191      PUSHL #13      ;
                                00000000G 9F 02 FB 00193      CALLS #2, a#DS$ENDSUB      ;
                                02 DD 0019A      PUSHL #2      ; 0964
                                0D DD 0019C      PUSHL #13      ;
                                00000000G 9F 02 FB 0019E      CALLS #2, a#DS$BGNSUB      ;
;
; Invalidate all map entries and initialize the UET data register.
;
50 0000G CF 0003F462 8F C9 001A5      BISL3 #259170, UNIBUS_SPACE, R0      ; 0973
                                60 01 AE 001AF      MNEGW #1, (R0)      ; 0974
53 01 1F 00  F0 001B2      INSV #0, #31, #1, MAP_ENTRY      ; 0975
                                50 D4 001B7      CLRL N      ; 0978
                                51 0000GDF40 DE 001B9 8$: MOVAL aUBI_UNDER_TEST[N], R1      ;
                                C1 0800 53 D0 001BF      MOVL MAP_ENTRY, 2048(R1)      ;
                                ED 50 000001FF 8F F3 001C4      AOBLEQ #511, N, 8$      ; 0976
;
; Write a unique data pattern to a memory location, set up and
; execute a
; DATI from that location, and check that the UET data register
; contains the correct data. If it does not, then the correct
; map has
; not been addressed properly due to a stuck or shorted Unibus
; page frame
; bit. This sequence is repeated for each map entry of the set
; of map
; entries 0, 1, 2, 4, 8, 16, 32, 64, 128, and 256.
;
                                04 AE 01 D0 001CC      MOVL #1, I      ; 0993
                                5B D4 001D0 9$: CLRL M      ; 0996
                                FF AB 9E 001D2 10$: MOVAB -1(R11), R0      ; 0999
                                50 50 78 001D6      ASHL R0, #1, N      ;
                                03 0000G CF 55 E0 001DA      BBS N, VALID_MAPS, 11$      ; 1000
                                0136 31 001E0      BRW 16$      ;
                                08 AE 04 AE 02 78 001E3 11$: ASHL #2, I, 8(SP)      ; 1006
                                50 0000G CF 08 AE C1 001E9 12$: ADDL3 8(SP), UBI_UNDER_TEST, R0      ;
                                60 01 D0 001F0      MOVL #1, (R0)      ;
;
; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.
;

```


56	07	09		55	F0 001F3	INSV	N, #9, #7, UBUS_ADDR	; 1017
		50	0000G	CF	9E 001F8	MOVAB	SCRATCH, R0	; 1018
56	09	00		50	F0 001FD	INSV	R0, #0, #9, UBUS_ADDR	; 1019
	50	0000G	CF	0003F460	8F C9 00202	BISL3	#259168, UNIBUS_SPACE, R0	; 1019
		60		56	B0 0020C	MOVW	UBUS_ADDR, (R0)	; 1019
; Write the two MSBs of the Unibus address with the two								
; MSBs of the map number.								
50	55	02		07	EF 0020F	EXTZV	#7, #2, N, R0	; 1030
	5A	50		08	AS 00214	MULW3	#8, R0, TEMP3	; 1030
; Set up the UBI map. Put the CMI page frame number								
; (PFN) into the register BUF_PFN. Then call the map								
; setup routine with the parameters map number, PFN of								
; the CMI buffer, byte offset mode (0 = no address								
; offset), and data path number.								
52	57	57	0000G	CF	9E 00218	MOVAB	SCRATCH, TEMP	; 1044
		0F		09	EF 0021D	EXTZV	#9, #15, TEMP, BUF_PFN	; 1045
			04	AE	DD 00222	PUSHL	I	; 1046
				7E	D4 00225	CLRL	-(SP)	; 1046
				52	DD 00227	PUSHL	BUF_PFN	; 1046
				55	DD 00229	PUSHL	N	; 1046
		0000G	CF	04	FB 0022B	CALLS	#4, MAP_SETUP	; 1047
		0000G	CF	5B	D0 00230	MOVL	M, SCRATCH	; 1047
; Execute a DATI and after a nominal wait compare the								
; data in the UET data register with the expected								
; data. If there is an error, report it.								
	50	0000G	CF	0003F464	8F C9 00235	BISL3	#259172, UNIBUS_SPACE, R0	; 1057
	60		5A		01 A9 0023F	BISW3	#1, TEMP3, (R0)	; 1059
			7E		0A 7D 00243	MOVQ	#10, -(SP)	; 1061
		00000000G	9F		02 FB 00246	CALLS	#2, a#DS\$WAITUS	; 1062
08	AE	04	AE	02	78 0024D	ASHL	#2, I, 8(SP)	; 1062
	50	0000G	CF	08	AE C1 00253	ADDL3	8(SP), UBI_UNDER_TEST, R0	; 1062
0000'	CF		60	1FFFFFFE	8F CB 0025A	BICL3	#536870910, (R0), STATUS1	; 1062
			50	0000'	CF D0 00264	MOVL	STATUS1, R0	; 1062
				21	18 00269	BGEQ	13\$	; 1062
				7E	7C 0026B	CLRQ	-(SP)	; 1062
				7E	D4 0026D	CLRL	-(SP)	; 1062
				50	DD 0026F	PUSHL	R0	; 1062
				7E	D4 00271	CLRL	-(SP)	; 1062
		18	AE	DD 00273	PUSHL	I		; 1062
		0000G	CF	9F 00276	PUSHAB	PRINT_CSR_ERR		; 1062
		0000G	CF	9F 0027A	PUSHAB	UBIMOD UB_CMI		; 1062
		0000G	CF	9A 0027E	MOVZBL	LUN, -(SP)		; 1062
			04	DD 00283	PUSHL	#4		; 1062
			0A	FB 00285	CALLS	#10, a#DS\$ERRHARD		; 1062
50		0000G	CF	0003F464	8F C9 0028C	BISL3	#259172, UNIBUS_SPACE, R0	; 1063
		0000'	CF		60 B0 00296	MOVW	(R0), STATUS0	; 1063
		0000'	CF	FF1E	8F AA 0029B	BICW2	#65310, STATUS0	; 1063
			50	0000'	CF 32 002A2	CVTWL	STATUS0, R0	; 1063
				21	13 002A7	BEQL	14\$	; 1063

Address	Op Code	Op Name	Comments	Address	Op Code	Op Name	Comments
5B	7E	0000G	0003F462	7E	7C	002A9	CLRQ -(SP)
58	9F	0000G	0003F462	50	DD	002AB	PUSHL R0
58	CF	0000G	0003F462	02	7D	002AD	MOVQ #2, -(SP)
58	00	0000G	0003F462	CF	9F	002B0	PUSHAB FMT_UET_STAT
58	10	0000G	0003F462	CF	9F	002B4	PUSHAB PRINT_GENERAL
58	23	0000G	0003F462	CF	9F	002B8	PUSHAB UBIMOD_UB_CMI
58	7E	0000G	0003F462	CF	9A	002BC	MOVZBL LUN, -(SP)
58	58	0000G	0003F462	05	DD	002C1	PUSHL #5
58	5B	0000G	0003F462	0A	FB	002C3	CALLS #10, a#DS\$ERRHARD
58	02	0000G	0003F462	8F	C9	002CA	BISL3 #259170, UNIBUS_SPACE, R0
58	06	0000G	0003F462	60	B0	002D4	MOVW (R0), TEMP1
58	00	0000G	0003F462	00	ED	002D7	CMPZV #0, #16, TEMP1, M
58	23	0000G	0003F462	13	00	002DC	BEQL 15\$
58	7E	0000G	0003F462	7C	00	002DE	CLRQ -(SP)
58	58	0000G	0003F462	3C	00	002E0	MOVZWL TEMP1, -(SP)
58	5B	0000G	0003F462	DD	00	002E3	PUSHL M
58	02	0000G	0003F462	DD	00	002E5	PUSHL #2
58	06	0000G	0003F462	CF	9F	002E7	PUSHAB FMT_UB_CMI
58	00	0000G	0003F462	CF	9F	002EB	PUSHAB PRINT_GENERAL
58	23	0000G	0003F462	CF	9F	002EF	PUSHAB UBIMOD_UB_CMI
58	7E	0000G	0003F462	CF	9A	002F3	MOVZBL LUN, -(SP)
58	58	0000G	0003F462	06	DD	002F8	PUSHL #6
58	5B	0000G	0003F462	0A	FB	002FA	CALLS #10, a#DS\$ERRHARD
58	02	0000G	0003F462	45	DE	00301	MOVAL aUBI_UNDER_TEST[N], R0
58	06	0000G	0003F462	53	D0	00307	MOVL MAP_ENTRY, 2048(R0)
58	00	0000G	0003F462	00	FB	0030C	CALLS #0, a#DS\$INLOOP
58	03	0000G	0003F462	50	E9	00313	BLBC R0, 16\$
58	09	0000G	0003F462	31	00	00316	BRW 12\$
58	03	0000G	0003F462	F1	00	00319	ACBL #9, #1, M, 10\$
58	02	0000G	0003F462	F1	00	0031F	ACBL #3, #1, I, 9\$
58	0D	0000G	0003F462	DD	00	00326	PUSHL #2
58	02	0000G	0003F462	DD	00	00328	PUSHL #13
58	03	0000G	0003F462	FB	00	0032A	CALLS #2, a#DS\$ENDSUB
58	0D	0000G	0003F462	DD	00	00331	PUSHL #3
58	02	0000G	0003F462	DD	00	00333	PUSHL #13
58	02	0000G	0003F462	FB	00	00335	CALLS #2, a#DS\$BGNSUB
58	01	0000G	0003F462	D0	00	0033C	MOVL #1, I
58	04	0000G	0003F462	AE	D4	0033F	CLRL M
58	02	0000G	0003F462	78	00	00342	ASHL #2, I, 8(SP)
58	08	0000G	0003F462	AE	C1	00347	ADDL3 8(SP), UBI_UNDER_TEST, R0
58	01	0000G	0003F462	D0	00	0034E	MOVL #1, (R0)

; Float a one through each byte number bit of the bits 2 through
 ; h 8 of
 ; each BAR. Set up a DATI and initiate it for each of the long
 ; word
 ; elements of the vector SCRATCH in the sequence 0, 1, 2, 4, 8,
 ; 16, 32,
 ; 64. After the DATI, change the contents of the CMI location,
 ; execute
 ; another DATI, and check that the contents of the UET data
 ; register did not change. If it did, there is a stuck or shor
 ; ted BAR
 ; bit. Repeat this sequence for each BDP.

```

; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.
;
56      07      09      54 F0 00351      INSV      MAP_NUMBER, #9, #7, UBUS_ADDR      ; 1122
50      04      AE      01 C3 00356      SUBL3     #1, M, R0      ; 1123
55      01      50      78 0035B      ASHL      R0, #1, N      ;
56      50      0000GCF45 DE 0035F      MOVAL     SCRATCH[N], R0      ; 1124
50      09      00      50 F0 00365      INSV      R0, #0, #9, UBUS_ADDR      ;
50      0000G CF 0003F460 8F C9 0036A      BISL3     #259168, UNIBUS_SPACE, R0      ;
50      60      56      B0 00374      MOVW      UBUS_ADDR, (R0)      ; 1125

; Write the two MSBs of the Unibus address with the two
; MSBs of the map number.
;
50      54      02      07 EF 00377      EXTZV     #7, #2, MAP_NUMBER, R0      ; 1136
50      5A      50      08 A5 0037C      MULW3     #8, R0, TEMP3      ;

; Set up the UBI map. Put the CMI page frame number
; (PFN) into the register BUF_PFN. Then call the map
; setup routine with the parameters map number, PFN of
; the CMI buffer, byte offset mode (0 = no address
; offset), and data path number.
;
52      57      57      0000G CF 9E 00380      MOVAB     SCRATCH, TEMP      ; 1150
50      0F      09      EF 00385      EXTZV     #9, #15, TEMP, BUF_PFN      ; 1151
50      5B      DD 0038A      PUSHL     I      ; 1152
50      7E      D4 0038C      CLRL     -(SP)      ;
50      52      DD 0038E      PUSHL     BUF_PFN      ;
50      54      DD 00390      PUSHL     MAP_NUMBER      ;
50      04      FB 00392      CALLS     #4, MAP_SETUP      ;
50      0000G CF 04 AE D0 00397      MOVL      M, SCRATCH[N]      ; 1153
50      0000GCF45      ;

; Execute a DATI and after a nominal wait, change the data in
; the CMI location and execute a second DATI. Then check
; the data in the UET data register did not change.
; If it did, report the error.
;
50      0000G CF 0003F464 8F C9 0039E      BISL3     #259172, UNIBUS_SPACE, R0      ; 1164
50      51      5A 3C 003A8      MOVZWL    TEMP3, R1      ; 1166
50      6E      01      C9 003AB      BISL3     #1, R1, (SP)      ;
50      60      6E B0 003AF      MOVW      (SP), (R0)      ;
50      7E      0A 7D 003B2      MOVQ      #10, -(SP)      ; 1168
50      00000000G 9F 02 FB 003B5      CALLS     #2, @DS$WAITUS      ;
50      AE      5B 02 78 003BC      ASHL      #2, I, 8(SP)      ; 1169
50      0000G CF 08 AE C1 003C1      ADDL3     8(SP), UBI_UNDER_TEST, R0      ;
50      0000' CF 60 1FFFFFFE 8F CB 003C8      BISL3     #536870910, (R0), STATUS1      ;
50      50      CF 0000'      MOVL      STATUS1, R0      ;
50      20      18 003D7      BGEQ      20$      ;
50      7E      7C 003D9      CLRL     -(SP)      ;
50      7E      D4 003DB      CLRL     -(SP)      ;
50      50      DD 003DD      PUSHL     R0      ;
50      7E      D4 003DF      CLRL     -(SP)      ;

```


			5B	DD	003E1	PUSHL	I	:
		0000G	CF	9F	003E3	PUSHAB	PRINT_CSR_ERR	:
		0000G	CF	9F	003E7	PUSHAB	UBIMOD_BYTE_OFF	:
	7E	0000G	CF	9A	003EB	MOVZBL	LUN, -(SP)	:
			07	DD	003F0	PUSHL	#7	:
50	00000000G	9F	0A	FB	003F2	CALLS	#10, a#DS\$ERRHARD	:
	0000G	CF	8F	C9	003F9	BISL3	#259172, UNIBUS_SPACE, R0	1170
	0000'	CF	60	B0	00403	MOVW	(R0), STATUS0	:
	0000'	CF	8F	AA	00408	BICW2	#65310, STATUS0	:
		50	CF	32	0040F	CVTL	STATUS0, R0	:
			21	13	00414	BEQL	21\$	:
			7E	7C	00416	CLRQ	-(SP)	:
			50	DD	00418	PUSHL	R0	:
		7E	02	7D	0041A	MOVQ	#2, -(SP)	:
			CF	9F	0041D	PUSHAB	FMT_UET_STAT	:
		0000G	CF	9F	00421	PUSHAB	PRINT_GENERAL	:
		0000G	CF	9F	00425	PUSHAB	UBIMOD_BYTE_OFF	:
		7E	CF	9A	00429	MOVZBL	LUN, -(SP)	:
			08	DD	0042E	PUSHL	#8	:
	00000000G	9F	0A	FB	00430	CALLS	#10, a#DS\$ERRHARD	:
	0000G	CF	AE	D2	00437	MCOML	M, SCRATCH[N]	1171
50	0000G	CF	8F	C9	0043E	BISL3	#259168, UNIBUS_SPACE, R0	:
		60	56	B0	00448	MOVW	UBUS_ADDR, (R0)	1172
50	0000G	CF	8F	C9	0044B	BISL3	#259172, UNIBUS_SPACE, R0	:
		60	6E	B0	00455	MOVW	(SP), (R0)	1174
		7E	0A	7D	00458	MOVQ	#10, -(SP)	1176
	00000000G	9F	02	FB	0045B	CALLS	#2, a#DS\$WAITUS	:
50	0000G	CF	AE	C1	00462	ADDL3	8(SP), UBI_UNDER_TEST, R0	1177
0000'		60	8F	CB	00469	BICL3	#536870910, (R0), STATUS1	:
		50	CF	D0	00473	MOVL	STATUS1, R0	:
			20	18	00478	BGEQ	22\$	:
			7E	7C	0047A	CLRQ	-(SP)	:
			7E	D4	0047C	CLRL	-(SP)	:
			50	DD	0047E	PUSHL	R0	:
			7E	D4	00480	CLRL	-(SP)	:
			5B	DD	00482	PUSHL	I	:
		0000G	CF	9F	00484	PUSHAB	PRINT_CSR_ERR	:
		0000G	CF	9F	00488	PUSHAB	UBIMOD_BYTE_OFF	:
		7E	CF	9A	0048C	MOVZBL	LUN, -(SP)	:
			09	DD	00491	PUSHL	#9	:
	00000000G	9F	0A	FB	00493	CALLS	#10, a#DS\$ERRHARD	:
50	0000G	CF	8F	C9	0049A	BISL3	#259172, UNIBUS_SPACE, R0	1178
	0000'	CF	60	B0	004A4	MOVW	(R0), STATUS0	:
	0000'	CF	8F	AA	004A9	BICW2	#65310, STATUS0	:
		50	CF	32	004B0	CVTL	STATUS0, R0	:
			21	13	004B5	BEQL	23\$	:
			7E	7C	004B7	CLRQ	-(SP)	:
			50	DD	004B9	PUSHL	R0	:
		7E	02	7D	004BB	MOVQ	#2, -(SP)	:
			CF	9F	004BE	PUSHAB	FMT_UET_STAT	:
		0000G	CF	9F	004C2	PUSHAB	PRINT_GENERAL	:
		0000G	CF	9F	004C6	PUSHAB	UBIMOD_BYTE_OFF	:
		7E	CF	9A	004CA	MOVZBL	LUN, -(SP)	:
			0A	DD	004CF	PUSHL	#10	:
	00000000G	9F	0A	FB	004D1	CALLS	#10, a#DS\$ERRHARD	:

```

50      0000G CF 0003F462 8F C9 004D8 23$: BISL3 #259170, UNIBUS_SPACE, R0 ; 1179
58      60 B0 004E2 MOVW (R0), TEMP1 ; 1180
59      0000GCF45 F7 004E5 CRTLW SCRATCH(N), TEMP2 ; 1181
59      58 B1 004EB CMPW TEMP1, TEMP2 ; 1187
24      12 004EE BNEQ 24$ ; 1193
7E      58 3C 004F2 CLRL -(SP) ; 1199
7E      59 3C 004F5 MOVZWL TEMP1, -(SP) ; 1200
02      DD 004F8 PUSHL #2 ; 1208
0000G CF 9F 004FA PUSHAB FMT_BYTE_OFF ; 1210
0000G CF 9F 004FE PUSHAB PRINT_GENERAL ; 1211
0000G CF 9F 00502 PUSHAB UBIMOD_BYTE_OFF ; 1212
7E      0000G CF 9A 00506 MOVZBL LUN, -(SP) ; 1215
0B      DD 0050B PUSHL #11 ; 1216
00000000G 9F 0A FB 0050D CALLS #10, a#DS$ERRHARD ; 1217
00000000G 9F 00 FB 00514 CALLS #0, a#DS$INLOOP ; 1218
03      50 E9 0051B BLBC R0, 25$ ; 1219
FE26 31 0051E BRW 19$ ; 1220
FE1A 04 AE 01 07 F1 00521 25$: ACBL #7, #1, M, 18$ ; 1232
FE11 5B 01 03 F1 00528 ACBL #3, #1, I, 17$ ; 1233
03      DD 0052E PUSHL #3 ; 1234
0D      DD 00530 PUSHL #13 ; 1235
00000000G 9F 02 FB 00532 CALLS #2, a#DS$ENDSUB ; 1236
04      DD 00539 PUSHL #4 ; 1237
0D      DD 0053B PUSHL #13 ; 1238
00000000G 9F 02 FB 0053D CALLS #2, a#DS$BGNSUB ; 1239
; Invalidate all map entries and initialize the UET data register.
;
50      0000G CF 0003F462 8F C9 00544 BISL3 #259170, UNIBUS_SPACE, R0 ; 1208
60      01 AE 0054E MNEGW #1, (R0) ; 1210
01      1F 00 F0 00551 INSV #0, #31, #1, MAP_ENTRY ; 1211
50      D4 00556 CLRL N ; 1215
51      0000GDF40 DE 00558 26$: MOVAL aUBI_UNDER_TEST(N), R1 ; 1212
C1      53 D0 0055E MOVL MAP_ENTRY, 2048(R1) ; 1212
ED      50 000001FF 8F F3 00563 AOBLEQ #511, N, 26$ ; 1212
; Write a unique data pattern to a memory location, set up and
; execute a
; DATI from that location, and check that the UET data register
; contains the correct data. If it does not, then the correct
; map has
; not been addressed properly due to a stuck or shorted Unibus
; page frame
; bit. This sequence is repeated for each map entry of the set
; of map
; entries 0, 1, 2, 4, 8, 16, 32, 64, 128, and 256.
;
59      01 D0 0056B MOVL #1, I ; 1232
54      D4 0056E 27$: CLRL M ; 1235
50      FF A4 9E 00570 28$: MOVAB -1(R4), R0 ; 1238
55      01 50 78 00574 ASHL R0, #1, N ; 1239
03      0000G CF 55 E0 00578 BBS N, VALID_MAPS, 29$ ; 1239
01D6 31 0057E BRW 36$ ;

```

```

08 AE 50 0000G 59 CF 60 08 AE 01 78 00581 29$: ASHL #2, I, 8(SP) ; 1245
                                30$: ADDL3 8(SP), UBI_UNDER_TEST, R0 ;
                                MOVL #1, (R0) ;
; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.
;
56 07 09 50 0000G 55 F0 00590 INSV N, #9, #7, UBUS_ADDR ; 1256
                                56 09 00 50 0000G 55 F0 00595 MOVAB SCRATCH, R0 ; 1257
                                56 50 0000G CF 0003F460 8F C9 0059A INSV R0, #0, #9, UBUS_ADDR ;
                                56 60 0003F460 8F C9 0059F BISL3 #259168, UNIBUS_SPACE, R0 ;
                                56 B0 005A9 MOVW UBUS_ADDR, (R0) ; 1258
; Write the two MSBs of the Unibus address with the two
; MSBs of the map number.
;
50 55 02 5A 50 07 EF 005AC EXTZV #7, #2, N, R0 ; 1269
                                08 A5 005B1 MULW3 #8, R0, TEMP3 ;
; Set up the UBI map. Put the CMI page frame number
; (PFN) into the register BUF_PFN. Then call the map
; setup routine with the parameters map number, PFN of
; the CMI buffer, byte offset mode (0 = no address
; offset), and data path number.
;
52 57 57 0F 0000G CF 9E 005B5 MOVAB SCRATCH, TEMP ; 1283
                                09 EF 005BA EXTZV #9, #15, TEMP, BUF_PFN ; 1284
                                59 DD 005BF PUSHL I ; 1285
                                7E D4 005C1 CLRL -(SP) ;
                                52 DD 005C3 PUSHL BUF_PFN ;
                                55 DD 005C5 PUSHL N ;
                                0000G CF 04 FB 005C7 CALLS #4, MAP_SETUP ;
                                0000G CF 54 D0 005CC MOVL M, SCRATCH ; 1286
; Execute a DATI and after a nominal wait, change the
; data in the CMI location and execute a second DATI.
; Then check the data in the UET data register
; did not change. If it did, report the error.
;
08 AE 50 0000G 59 CF 60 08 AE 01 78 00581 29$: ASHL #2, I, 8(SP) ; 1297
                                30$: ADDL3 8(SP), UBI_UNDER_TEST, R0 ; 1299
                                MOVL #1, (R0) ;
                                MOVW R11, (R0) ;
                                MOVQ #10, -(SP) ; 1301
                                CALLS #2, @DS$WAITUS ;
                                ASHL #2, I, 8(SP) ; 1302
                                ADDL3 8(SP), UBI_UNDER_TEST, R0 ;
                                BISL3 #536870910, (R0), STATUS1 ;
                                MOVL STATUS1, R0 ;
                                BGEQ 31$ ;
                                CLRL -(SP) ;
                                CLRL -(SP) ;
                                PUSHL R0 ;

```


			7E	D4	00611	CLRL	-(SP)	:	
			59	DD	00613	PUSHL	I	:	
		0000G	CF	9F	00615	PUSHAB	PRINT_CSR_ERR	:	
		0000G	CF	9F	00619	PUSHAB	UBIMOD_UB_CMI	:	
		0000G	CF	9A	0061D	MOVZBL	LUN, -(SP)	:	
			0C	DD	00622	PUSHL	#12	:	
	00000000G	9F	0A	FB	00624	CALLS	#10, a#DS\$ERRHARD	:	
50	0000G	CF	0003F464	8F	C9	0062B	31\$: BISL3	:	1303
	0000'	CF		60	B0	00635		:	
	0000'	CF	FF1E	8F	AA	0063A		:	
		50	0000'	CF	32	00641		:	
				21	13	00646		:	
				7E	7C	00648		:	
				50	DD	0064A		:	
		7E		02	7D	0064C		:	
		0000G		CF	9F	0064F		:	
		0000G		CF	9F	00653		:	
		0000G		CF	9F	00657		:	
		7E	0000G	CF	9A	0065B		:	
				0D	DD	00660		:	
	00000000G	9F	0A	FB	00662	CALLS	#10, a#DS\$ERRHARD	:	
	0000G	CF		54	D2	00669	32\$: MCOML	:	1304
50	0000G	CF	0003F460	8F	C9	0066E		:	
		60		56	B0	00678		:	1305
50	0000G	CF	0003F464	8F	C9	0067B		:	
		60		5B	B0	00685		:	1307
		7E		0A	7D	00688		:	1309
	00000000G	9F		02	FB	0068B		:	
	50	CF	08	AE	C1	00692		:	1310
0000'		60	1FFFFFFE	8F	CB	00699		:	
		50	0000'	CF	D0	006A3		:	
				20	18	006A8		:	
				7E	7C	006AA		:	
				7E	D4	006AC		:	
				50	DD	006AE		:	
				7E	D4	006B0		:	
				59	DD	006B2		:	
		0000G		CF	9F	006B4		:	
		0000G		CF	9F	006B8		:	
		7E	0000G	CF	9A	006BC		:	
				0E	DD	006C1		:	
	00000000G	9F	0A	FB	006C3	CALLS	#10, a#DS\$ERRHARD	:	
50	0000G	CF	0003F464	8F	C9	006CA	33\$: BISL3	:	1311
	0000'	CF		60	B0	006D4		:	
	0000'	CF	FF1E	8F	AA	006D9		:	
		50	0000'	CF	32	006E0		:	
				21	13	006E5		:	
				7E	7C	006E7		:	
				50	DD	006E9		:	
		7E		02	7D	006EB		:	
		0000G		CF	9F	006EE		:	
		0000G		CF	9F	006F2		:	
		0000G		CF	9F	006F6		:	
		7E	0000G	CF	9A	006FA		:	
				0F	DD	006FF		:	

						CLRL	-(SP)	:	
						PUSHL	I	:	
						PUSHAB	PRINT_CSR_ERR	:	
						PUSHAB	UBIMOD_UB_CMI	:	
						MOVZBL	LUN, -(SP)	:	
						PUSHL	#12	:	
						CALLS	#10, a#DS\$ERRHARD	:	
						BISL3	#259172, UNIBUS_SPACE, R0	:	1303
						MOVW	(R0), STATUS0	:	
						BICW2	#65310, STATUS0	:	
						CVTWL	STATUS0, R0	:	
						BEQL	32\$	:	
						CLRQ	-(SP)	:	
						PUSHL	R0	:	
						MOVQ	#2, -(SP)	:	
						PUSHAB	FMT_UET_STAT	:	
						PUSHAB	PRINT_GENERAL	:	
						PUSHAB	UBIMOD_UB_CMI	:	
						MOVZBL	LUN, -(SP)	:	
						PUSHL	#13	:	
						CALLS	#10, a#DS\$ERRHARD	:	
						MCOML	M, SCRATCH	:	1304
						BISL3	#259168, UNIBUS_SPACE, R0	:	
						MOVW	UBUS_ADDR, (R0)	:	1305
						BISL3	#259172, UNIBUS_SPACE, R0	:	
						MOVW	R11, (R0)	:	1307
						MOVQ	#10, -(SP)	:	1309
						CALLS	#2, a#DS\$WAITUS	:	
						ADDL3	8(SP), UBI_UNDER_TEST, R0	:	1310
						BICL3	#536870910, (R0), STATUS1	:	
						MOVL	STATUS1, R0	:	
						BGEQ	33\$	:	
						CLRQ	-(SP)	:	
						CLRL	-(SP)	:	
						PUSHL	R0	:	
						CLRL	-(SP)	:	
						PUSHL	I	:	
						PUSHAB	PRINT_CSR_ERR	:	
						PUSHAB	UBIMOD_UB_CMI	:	
						MOVZBL	LUN, -(SP)	:	
						PUSHL	#14	:	
						CALLS	#10, a#DS\$ERRHARD	:	
						BISL3	#259172, UNIBUS_SPACE, R0	:	1311
						MOVW	(R0), STATUS0	:	
						BICW2	#65310, STATUS0	:	
						CVTWL	STATUS0, R0	:	
						BEQL	34\$	:	
						CLRQ	-(SP)	:	
						PUSHL	R0	:	
						MOVQ	#2, -(SP)	:	
						PUSHAB	FMT_UET_STAT	:	
						PUSHAB	PRINT_GENERAL	:	
						PUSHAB	UBIMOD_UB_CMI	:	
						MOVZBL	LUN, -(SP)	:	
						PUSHL	#15	:	

ZZ-ECCBA-1.4
UBI_TESTS_11_15
01-04

TEST_013: Buffered Address Register Test
TEST_013: Buffered Address Register Test

J 7

6-Sep-1985

Fiche 2 Frame J7

Sequence 293

6-Sep-1985 17:19:55

VAX-11 Bliss-32 V4.1-746

Page 47

6-Sep-1985 17:14:57

[LEMIEUX.ECCBA.RELEASE]ECCBA5.B32;5

(5)

		00000000G	9F	0A	FB	00701	CALLS	#10, @DS\$ERRHARD	:		
	50	0000G	CF	0003F462	8F	C9 00708	34\$: BISL3	#259170, UNIBUS_SPACE, R0	:	1312	
			58		60	B0 00712	MOVW	(R0), TEMP1	:		
54	58		10		00	ED 00715	CMPZV	#0, #16, TEMP1, M	:	1313	
					23	13 0071A	BEQL	35\$	:		
					7E	7C 0071C	CLRQ	-(SP)	:	1319	
			7E		58	3C 0071E	MOVZWL	TEMP1, -(SP)	:		
					54	DD 00721	PUSHL	M	:		
					02	DD 00723	PUSHL	#2	:		
		0000G	CF		9F	00725	PUSHAB	FMT_UB_CMI	:		
		0000G	CF		9F	00729	PUSHAB	PRINT_GENERAL	:		
		0000G	CF		9F	0072D	PUSHAB	UBIMOD_UB_CMI	:		
			7E		0000G	CF	9A 00731	MOVZBL	LUN, -(SP)	:	
					10	DD 00736	PUSHL	#16	:		
		00000000G	9F		0A	FB 00738	CALLS	#10, @DS\$ERRHARD	:		
	50			0000G	DF45	DE 0073F	35\$: MOVAL	@UBI_UNDER_TEST(N), R0	:	1321	
		0800	C0		53	D0 00745	MOVL	MAP_ENTRY, 2048(R0)	:		
		00000000G	9F		00	FB 0074A	CALLS	#0, @DS\$INLOOP	:	1326	
			03		50	E9 00751	BLBC	R0, 36\$	:		
				FE2F	31	00754	BRW	30\$	:		
FE13	54		01		09	F1 00757	36\$: ACBL	#9, #1, M, 28\$	:	1235	
FE0B	59		01		03	F1 0075D	ACBL	#3, #1, I, 27\$	:	1232	
					04	DD 00763	PUSHL	#4	:	1334	
					0D	DD 00765	PUSHL	#13	:		
		00000000G	9F		02	FB 00767	CALLS	#2, @DS\$ENDSUB	:		
		00000000G	9F		00	FB 0076E	CALLS	#0, @DS\$BREAK	:	1335	
			50		01	D0 00775	MOVL	#1, R0	:	1337	
					04	00778	RET		:		

; Routine Size: 1913 bytes, Routine Base: TEST_013 + 0000

\$DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Buffered Data Path DATI Test');

!++
! TEST DESCRIPTION:

This tests the ability of the UBI to perform DATI transactions through the buffered data paths. The following data patterns are used to verify the integrity of the data paths:

pattern 1	01010101010101010101010101010101
pattern 2	10101010101010101010101010101010
pattern 3	00110011001100110011001100110011
pattern 4	00001111000011110000111100001111
pattern 5	00000000111111110000000011111111

The following algorithm is used. The UET data register is cleared, then a DATI is executed with address bit 1 at a zero. After a nominal wait, the data in the UET data register is verified. Then the data pattern in memory is changed. ! A11
Then a second DATI is executed with address bit 1 at a one, and the data in the UET data register is verified again. This is repeated for each BDP, and for longword and word aligned CMI references.

! ASSUMPTIONS:

! Test 1-Test 13

REGISTER

BUF_PFN,
M,
MAP_NUMBER,
N,
TEMP,
TEMP1:WORD,
TEMP3:WORD,
UBUS_ADDR: WORD;

MACRO

UBUS_PF = UBUS_ADDR<9,7>X, ! Defines the Unibus page frame field
UBUS_BO = UBUS_ADDR<0,9>X; ! Defines the Unibus byte number field

MAP

SCRATCH: VECTOR[5,WORD];

CODECOMMENT

'Call the Unibus sizer routine to locate any Unibus memory. This builds',
'a table of useable map entries. In the test which follows, the map',
'selected is drawn from this table.'

BEGIN

UNIBUS_SIZER();


```

: 1393 3
: 1394 2      END;
: 1395 2
: 1396 2      CODECOMMENT
: 1397 2      ''
: 1398 2      'Load the data patterns into the table WORD_PATTERN for later use.',
: 1399 2      ''
: 1400 3      BEGIN
: 1401 3
: 1402 3      WORD_PATTERN[0] = %X'5555';
: 1403 3      WORD_PATTERN[1] = %X'AAAA';
: 1404 3      WORD_PATTERN[2] = %X'3333';
: 1405 3      WORD_PATTERN[3] = %X'0F0F';
: 1406 3      WORD_PATTERN[4] = %X'00FF';
: 1407 3
: 1408 2      END;
: 1409 2
: 1410 2      CODECOMMENT
: 1411 2      ''
: 1412 2      'Set up the UBI and UET for a DATI, and set',
: 1413 2      'up the CMI source location (longword aligned for M = 0, word aligned',
: 1414 2      'for M = 1) with the appropriate data pattern. Execute the DATI and',
: 1415 2      'check that the UET data register contains the data pattern.',
: 1416 2      'Execute the second DATI with the Unibus address bit 1 at a one and',
: 1417 2      'check the UET data register again. Repeat this sequence for',
: 1418 2      'each BDP, for each data pattern, and for each variation in alignment.',
: 1419 2      ''
: 1420 3      BEGIN
: 1421 3
: 1422 3      INCR I FROM 1 TO 3 DO
: 1423 4          BEGIN
: 1424 4
: 1425 4          INCR M FROM 0 TO 1 DO
: 1426 5              BEGIN
: 1427 5
: 1428 5              INCR N FROM 0 TO 4 DO
: 1429 6                  BEGIN
: 1430 6
: 1431 6                  DO                                ! Scope loop starts here
: 1432 7                      BEGIN
: 1433 7
: 1434 7                      UBI_CSR(.I) = PURGE;
: 1435 7                      SCRATCH(.M) = .WORD_PATTERN[.N];
: 1436 7                      SCRATCH(.M + 1) = .WORD_PATTERN[.N];
: 1437 7
: 1438 7                      CODECOMMENT
: 1439 7                      ''
: 1440 7                      'Set up the UET address register with the Unibus',
: 1441 7                      'address. Bits 9 to 15 (the Unibus page frame) contain',
: 1442 7                      'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
: 1443 7                      'the byte number field of the CMI address.',
: 1444 7                      ''
: 1445 8                      BEGIN
: 1446 8
: 1447 8                      UET_DATA_REG = 0;

```

```

; 1448 8      MAP_NUMBER = .FREE_MAP[.I];
; 1449 8      UBUS_PF = .MAP_NUMBER;
; 1450 8      UBUS_BO = SCRATCH[.M];
; 1451 8      UET_ADDR_REG = .UBUS_ADDR;
; 1452 8
; 1453 7      END;
; 1454 7
; 1455 7      CODECOMMENT
; 1456 7      'Write the 2 MSBs of the Unibus address with the 2 MSBs',
; 1457 7      'of the map number.',
; 1458 7      '
; 1459 7      '
; 1460 8      BEGIN
; 1461 8
; 1462 8          TEMP3 = .MAP_NUMBER<7,2> * 8;
; 1463 8
; 1464 7      END;
; 1465 7
; 1466 7      CODECOMMENT
; 1467 7      'Set up the UBI map. Put the CMI page frame number',
; 1468 7      '(PFN) into the register BUF_PFN. Then call the map',
; 1469 7      'setup routine with the parameters map number, PFN of',
; 1470 7      'the CMI buffer, no address offset, and data path number',
; 1471 7      '
; 1472 7      '
; 1473 8      BEGIN
; 1474 8
; 1475 8          TEMP = SCRATCH[.M];
; 1476 8          BUF_PFN = .TEMP<9,15>;
; 1477 8          MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.I);
; 1478 8
; 1479 7      END;
; 1480 7
; 1481 7      CODECOMMENT
; 1482 7      'Execute the DATI and after a nominal wait compare the',
; 1483 7      'contents of the UET data register with the',
; 1484 7      'expected value. If there is an error, report it.',
; 1485 7      '
; 1486 7      '
; 1487 8      BEGIN
; 1488 8
; 1489 8          UET_CTRL_REG = .TEMP3 OR DATI;
; 1490 8
; 1491 8          $DS_WAITUS(TIME=10);
; L 1492 8          UBI_STATUS(.I,UBIMOD_DDP);          ! M7
; ;PRINT:      Test 14, Subtest 0, Error 1
; L 1493 9          UET_STATUS(UBIMOD_DDP);          ! M7
; ;PRINT:      Test 14, Subtest 0, Error 2
; 1494 8          TEMP1 = .UET_DATA_REG;
; 1495 8          IF .TEMP1 NEQ .WORD_PATTERN[.N]
; 1496 8          THEN
; 1497 9              BEGIN
; P 1498 9                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
; P 1499 9                      PRLINK = PRINT_GENERAL,
; P 1500 9                      P1 = FMT_BDP_DATI,P2 = 3,

```

```

; P 1501 9
; L 1502 9
; xPRINT:
; 1503 8
; 1504 8
; 1505 7
; 1506 7
; 1507 7
; 1508 7
; 1509 7
; 1510 7
; 1511 7
; 1512 7
; 1513 8
; 1514 8
; 1515 8
; 1516 8
; 1517 8
; 1518 9
; 1519 9
; 1520 9
; 1521 8
; 1522 8
; 1523 7
; 1524 7
; 1525 7
; 1526 7
; 1527 7
; 1528 7
; 1529 7
; 1530 7
; 1531 8
; 1532 8
; 1533 8
; 1534 8
; 1535 8
; 1536 8
; 1537 8
; L 1538 8
; xPRINT:
; L 1539 9
; xPRINT:
; 1540 8
; 1541 8
; 1542 8
; 1543 9
; P 1544 9
; P 1545 9
; P 1546 9
; P 1547 9
; L 1548 9
; xPRINT:
; 1549 8
; 1550 8
; 1551 7

Test 14, Subtest 0, Error 3
END;

CODECOMMENT
'Change the Unibus address bit 1 from a zero to a one.',
'Also change the data in memory if first reference was', ! A11
'on a long word boundry.', ! A11
';
BEGIN
UBUS_ADDR = .UBUS_ADDR + 2;
IF .M EQL 0
THEN
BEGIN
SCRATCH[.M] = 0;
SCRATCH[.M + 1] = 0;
END;
END;

CODECOMMENT
'Now execute another DATI, and check again that the',
'contents of the UET data register are as',
'expected',
';
BEGIN
UET_ADDR_REG = .UBUS_ADDR;
UET_CTRL_REG = .TEMP3 OR DATI;
$DS_WAITUS(TIME=10);
UBI_STATUS(.I,UBIMOD_DDP); ! M7
UET_STATUS(UBIMOD_DDP); ! M7
TEMP1 = .UET_DATA_REG;
IF .TEMP1 NEQ .WORD_PATTERN[.N]
THEN
BEGIN
$DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
PRLINK = PRINT_GENERAL,
P1 = FMT_BDP_DATI,P2 = 3,
P3 = .I,P4 = .WORD_PATTERN[.N],
P5 = .TEMP1);
END;
END;

Test 14, Subtest 0, Error 4
Test 14, Subtest 0, Error 5
Test 14, Subtest 0, Error 6
END;
END;

```



```
; 1552 7
; 1553 7
; 1554 6      END
; 1555 6      WHILE $DS_INLOOP;      ! Scope loop ends here
; 1556 5      END;
; 1557 5
; 1558 4      END;
; 1559 4
; 1560 3      END;
; 1561 3
; 1562 2      END;
; 1563 2
; 1564 1      $DS_ENDTEST;
```

```
.PSECT DISPATCH,NOWRT,NOEXE,2
00000000' 00000000' 00000000' 00000000' 00048 $: .ADDRESS P.AAJ, P.AAK, P.AAL, TEST_014
00000000 00000003 00058 .LONG 3, 0
.PSECT $PLIT$,NOWRT,NOEXE,2
20 61 74 61 44 20 64 65 72 65 66 66 75 42 1C 000E 00074 P.AAJ: .WORD 14
74 73 65 54 20 49 54 41 44 20 68 74 61 50 00076 P.AAK: .ASCII <28>\Buffered Data Path DATI Test\
00085
00093 .BLKB 1
00000000 00094 P.AAL: .LONG 0
.PSECT TEST_014,NOWRT,2
OFFC 00000 .ENTRY TEST_014, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1338
SE 04 C2 00002 R11
SUBL2 #4, SP
; Call the Unibus sizer routine to locate any Unibus memory. T
; his builds
; a table of useable map entries. In the test which follows, t
; he map
; selected is drawn from this table.
0000G CF 00 FB 00005 CALLS #0, UNIBUS_SIZER ; 1392
; Load the data patterns into the table WORD_PATTERN for later
; use.
0000' CF AAAA5555 8F D0 0000A MOVL #-1431677611, WORD_PATTERN ; 1402
0000' CF 0F0F3333 8F D0 00013 MOVL #252654387, WORD_PATTERN+4 ; 1404
0000' CF FF 8F 9B 0001C MOVZBW #255, WORD_PATTERN+8 ; 1406
; Set up the UBI and UET for a DATI, and set
; up the CMI source location (longword aligned for M = 0, word
; aligned
```

Line	Label	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419	Op420	Op421	Op422	Op423	Op424	Op425	Op426	Op427	Op428	Op429	Op430	Op431	Op432	Op433	Op434	Op435	Op436	Op437	Op438	Op439	Op440	Op441	Op442	Op443	Op444	Op445	Op446	Op447	Op448	Op449	Op450	Op451	Op452	Op453	Op454	Op455	Op456	Op457	Op458	Op459	Op460	Op461	Op462	Op463	Op464	Op46
------	-------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	------

```

50      0000G CF 0003F464 8F C9 00098      BISL3      #259172, UNIBUS_SPACE, R0      : 1487
60      56      01 A9 000A2      BISH3      #1, TEMP3, (R0)      : 1489
      7E      0A 7D 000A6      MOVQ      #10, -(SP)      : 1491
      00000000G 9F      02 FB 000A9      CALLS      #2, @DS$WAITUS      :
0000' CF 0000GDF48 1FFFFFFE 8F CB 000B0      BICL3      #536870910, @UBI_UNDER_TEST[1], STATUS1 : 1492
      50      0000' CF D0 000BD      MOVL      STATUS1, R0      :
      20 18 000C2      BGEQ      5$      :
      7E 7C 000C4      CLRQ      -(SP)      :
      7E D4 000C6      CLRL      -(SP)      :
      50 DD 000C8      PUSHL     R0      :
      7E D4 000CA      CLRL      -(SP)      :
      58 DD 000CC      PUSHL     I      :
      0000G CF 9F 000CE      PUSHAB    PRINT_CSR_ERR      :
      0000G CF 9F 000D2      PUSHAB    UBIMOD_DDP      :
      7E 0000G CF 9A 000D6      MOVZBL    LUN, -(SP)      :
      01 DD 000DB      PUSHL     #1      :
      00000000G 9F      0A FB 000DD      CALLS      #10, @DS$ERRHARD      :
50      0000G CF 0003F464 8F C9 000E4 5$: BISL3      #259172, UNIBUS_SPACE, R0      : 1493
      0000' CF 60 B0 000EE      MOVW      (R0), STATUS0      :
      0000' CF FF1E 8F AA 000F3      BICW2      #65310, STATUS0      :
      50      0000' CF 32 000FA      CVTWL     STATUS0, R0      :
      21 13 000FF      BEQL      6$      :
      7E 7C 00101      CLRQ      -(SP)      :
      50 DD 00103      PUSHL     R0      :
      7E      02 7D 00105      MOVQ      #2, -(SP)      :
      0000G CF 9F 00108      PUSHAB    FMT_UET_STAT      :
      0000G CF 9F 0010C      PUSHAB    PRINT_GENERAL      :
      0000G CF 9F 00110      PUSHAB    UBIMOD_DDP      :
      7E 0000G CF 9A 00114      MOVZBL    LUN, -(SP)      :
      02 DD 00119      PUSHL     #2      :
      00000000G 9F      0A FB 0011B      CALLS      #10, @DS$ERRHARD      :
50      0000G CF 0003F462 8F C9 00122 6$: BISL3      #259170, UNIBUS_SPACE, R0      : 1494
      55      60 B0 0012C      MOVW      (R0), TEMP1      :
      0000'CF4A 55 B1 0012F      CMPW      TEMP1, WORD_PATTERN[N] : 1495
      29 13 00135      BEQL      7$      :
      7E D4 00137      CLRL      -(SP)      : 1502
      55 3C 00139      MOVZWL    TEMP1, -(SP)      :
      7E 0000'CF4A 3C 0013C      MOVZWL    WORD_PATTERN[N], -(SP) :
      58 DD 00142      PUSHL     I      :
      03 DD 00144      PUSHL     #3      :
      0000G CF 9F 00146      PUSHAB    FMT_BDP_DATI      :
      0000G CF 9F 0014A      PUSHAB    PRINT_GENERAL      :
      0000G CF 9F 0014E      PUSHAB    UBIMOD_DDP      :
      7E 0000G CF 9A 00152      MOVZBL    LUN, -(SP)      :
      03 DD 00157      PUSHL     #3      :
      00000000G 9F      0A FB 00159      CALLS      #10, @DS$ERRHARD      :

;
; Change the Unibus address bit 1 from a zero to a one.
; Also change the data in memory if first reference was
; on a long word boundry.
;
57      02 A0 00160 7$: ADDW2      #2, UBUS_ADDR      : 1515
      59 D5 00163      TSTL      M      : 1516
      0A 12 00165      BNEQ      8$      :

```



```

                                0000GCF49 B4 00167          CLRW SCRATCH[M]          ; 1519
                                0000GCF49 B4 0016C          CLRW SCRATCH+2[M]        ; 1520
                                ;
                                ; Now execute another DATI, and check again that the
                                ; contents of the UET data register are as
                                ; expected
                                ;
50      0000G CF 0003F460      8F C9 00171 8$: BISL3 #259168, UNIBUS_SPACE, R0      ; 1531
60      0000G CF 0003F464      57 B0 0017B      MOVW UBUS_ADDR, (R0)                ; 1533
50      0000G CF 0003F464      8F C9 0017E      BISL3 #259172, UNIBUS_SPACE, R0      ;
60      0000G CF 0003F464      01 A9 00188      BISH3 #1, TEMP3, (R0)                ; 1535
6E      0000G CF 0003F464      0A 7D 0018C      MOVQ #10, -(SP)                      ; 1537
50      0000G CF 0003F464      02 FB 0018F      CALLS #2, @DS$WAITUS                    ;
50      0000G CF 0003F464      02 78 00196      ASHL #2, I, (SP)                      ; 1538
50      0000G CF 0003F464      6E C1 0019A      ADDL3 (SP), UBI_UNDER_TEST, R0        ;
0000' CF 0003F464      8F CB 001A0      BICL3 #536870910, (R0), STATUS1          ;
50      0000G CF 0003F464      20 18 001AF      MOVL STATUS1, R0                      ;
7E      0000G CF 0003F464      7E 7C 001B1      BGEQ 9$                                     ;
7E      0000G CF 0003F464      7E D4 001B3      CLRQ -(SP)                               ;
50      0000G CF 0003F464      50 DD 001B5      CLRL -(SP)                               ;
7E      0000G CF 0003F464      7E D4 001B7      PUSHL R0                                ;
58      0000G CF 0003F464      58 DD 001B9      CLRL -(SP)                               ;
0000G CF 0003F464      9F 9F 001BB      PUSHL I                                ;
0000G CF 0003F464      9F 9F 001BF      PUSHAB PRINT_CSR_ERR                    ;
7E      0000G CF 0003F464      9A 001C3      PUSHAB UBIMOD_DDP                        ;
04      0000G CF 0003F464      04 DD 001C8      MOVZBL LUN, -(SP)                        ;
9F      0000G CF 0003F464      0A FB 001CA      PUSHL #4                                ;
50      0000G CF 0003F464      8F C9 001D1 9$: CALLS #10, @DS$ERRHARD                ;
0000' CF 0003F464      60 B0 001DB      BISL3 #259172, UNIBUS_SPACE, R0          ; 1539
0000' CF 0003F464      8F AA 001E0      MOVW (R0), STATUS0                      ;
50      0000G CF 0003F464      32 001E7      BICW2 #65310, STATUS0                    ;
21      0000G CF 0003F464      13 001EC      CVTWL STATUS0, R0                        ;
7E      0000G CF 0003F464      7E 7C 001EE      BEQL 10$                                ;
50      0000G CF 0003F464      50 DD 001F0      CLRQ -(SP)                               ;
7E      0000G CF 0003F464      02 7D 001F2      PUSHL R0                                ;
0000G CF 0003F464      9F 9F 001F5      MOVQ #2, -(SP)                          ;
0000G CF 0003F464      9F 9F 001F9      PUSHAB FMT_UET_STAT                     ;
0000G CF 0003F464      9F 9F 001FD      PUSHAB PRINT_GENERAL                     ;
7E      0000G CF 0003F464      9A 00201      PUSHAB UBIMOD_DDP                        ;
05      0000G CF 0003F464      05 DD 00206      MOVZBL LUN, -(SP)                        ;
9F      0000G CF 0003F464      0A FB 00208      PUSHL #5                                ;
50      0000G CF 0003F464      8F C9 0020F 10$: CALLS #10, @DS$ERRHARD                ;
55      0000G CF 0003F464      60 B0 00219      BISL3 #259170, UNIBUS_SPACE, R0          ; 1540
5B      0000' CF 0003F464      3E 0021C      MOVW (R0), TEMP1                         ;
6B      0000' CF 0003F464      55 B1 00222      MOVAW WORD_PATTERN[N], R11              ;
26      0000G CF 0003F464      13 00225      CMPW TEMP1, (R11)                        ;
7E      0000G CF 0003F464      7E D4 00227      BEQL 11$                                ;
55      0000G CF 0003F464      3C 00229      CLRL -(SP)                               ;
7E      0000G CF 0003F464      6B 3C 0022C      MOVZWL TEMP1, -(SP)                      ;
58      0000G CF 0003F464      58 DD 0022F      MOVZWL (R11), -(SP)                      ;
03      0000G CF 0003F464      03 DD 00231      PUSHL I                                ;
0000G CF 0003F464      9F 9F 00233      PUSHL #3                                ;
0000G CF 0003F464      9F 9F 00237      PUSHAB FMT_BDP_DATI                     ;
0000G CF 0003F464      9F 9F 0023B      PUSHAB PRINT_GENERAL                     ;
                                PUSHAB UBIMOD_DDP                    ;

```

ZZ-ECCBA-1.4
UBI_TESTS_11_15
01-04

TEST_014: Buffered Data Path DATI Test
TEST_014: Buffered Data Path DATI Test

F 8

6-Sep-1985

Fiche 2 Frame F8

Sequence 302

6-Sep-1985 17:19:55

VAX-11 Bliss-32 V4.1-746

Page 56

6-Sep-1985 17:14:57

[LEMIEUX.ECCBA.RELEASE]ECCBA5.B32;5

(6)

		7E	0000G	CF	9A	0023F	MOVZBL	LUN, -(SP)	:
				06	DD	00244	PUSHL	#6	:
		00000000G	9F	0A	FB	00246	CALLS	#10, a#DS\$ERRHARD	:
		00000000G	9F	00	FB	0024D	CALLS	#0, a#DS\$INLOOP	: 1554
		03		50	E9	00254	BLBC	R0, 12\$	:
				FDD9	31	00257	BRW	4\$	:
FDC9	5A	01		04	F1	0025A	ACBL	#4, #1, N, 3\$	: 1428
FDC1	59	01		01	F1	00260	ACBL	#1, #1, M, 2\$	: 1425
FDB9	58	01		03	F1	00266	ACBL	#3, #1, I, 1\$	: 1422
		00000000G	9F	00	FB	0026C	CALLS	#0, a#DS\$BREAK	: 1562
			50	01	D0	00273	MOVL	#1, R0	: 1564
					04	00276	RET		:

; Routine Size: 631 bytes, Routine Base: TEST_014 + 0000

```

$DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Buffered Data Path DATIP Test');

!+
TEST DESCRIPTION:

This tests the ability of the UBI to perform a DATIP transaction
through the three BDPs. Actually, there should be no difference in
performance here from a DATI.

Before testing starts, the Unibus Sizer routine is called to determine
which maps are useable.

ASSUMPTIONS:

Test 1-Test 14

--

REGISTER
BUF_PFN,
M,
MAP_NUMBER,
N,
TEMP,
TEMP1:WORD,
TEMP2:WORD,
TEMP3,
UBUS_ADDR: WORD;

BUILTIN
MTPR;

MACRO
UBUS_PF = UBUS_ADDR<9,7>x,      ! Defines the Unibus page frame field
UBUS_BO = UBUS_ADDR<0,9>x;      ! Defines the Unibus byte number field

MAP
SCRATCH: VECTOR[5,WORD];

CODECOMMENT
''
'Call the Unibus sizer routine to locate any Unibus memory. This builds',
'a table of useable map entries. In the test which follows, the map',
'selected is drawn from this table.',
''
BEGIN
UNIBUS_SIZER();
END;

CODECOMMENT
''
'Set up the UBI UET for a DATIP, and',
'set up the CMI source location. Execute the DATIP and check that the',
'UET data register contains the data pattern.',

```



```

: 1620 2  '': BEGIN
: 1621 3
: 1622 3
: 1623 3   INCR I FROM 1 TO 3 DO
: 1624 4     BEGIN
: 1625 4
: 1626 4     DO                                ! Scope loop begins here
: 1627 5       BEGIN
: 1628 5
: 1629 5       UBI_CSR(.I) = PURGE;
: 1630 5       CODECOMMENT
: 1631 5       '':
: 1632 5       'Set up the UET address register with the Unibus',
: 1633 5       'address. Bits 9 to 15 (the Unibus page frame) contain',
: 1634 5       'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
: 1635 5       'the byte number field of the CMI address.',
: 1636 5       '':
: 1637 6         BEGIN
: 1638 6
: 1639 6         SCRATCH[0] = &X'1234';
: 1640 6         UET_DATA_REG = 0;
: 1641 6         MAP_NUMBER = .FREE_MAP(.I);
: 1642 6         UBUS_PF = .MAP_NUMBER;
: 1643 6         UBUS_BO = SCRATCH[0];
: 1644 6         UET_ADDR_REG = .UBUS_ADDR;
: 1645 6
: 1646 5         END;
: 1647 5
: 1648 5       CODECOMMENT
: 1649 5       '':
: 1650 5       'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
: 1651 5       'map number.',
: 1652 5       '':
: 1653 6         BEGIN
: 1654 6
: 1655 6         TEMP3 = .MAP_NUMBER<7,2> * 8;
: 1656 6
: 1657 5         END;
: 1658 5
: 1659 5       CODECOMMENT
: 1660 5       '':
: 1661 5       'Set up the UBI map. Put the CMI page frame number (PFN) into',
: 1662 5       'the register BUF_PFN. Then call the map setup routine with the',
: 1663 5       'parameters map number, PFN of the CMI buffer, no address offset',
: 1664 5       'and data path number.',
: 1665 5       '':
: 1666 6         BEGIN
: 1667 6
: 1668 6         TEMP = SCRATCH[0];
: 1669 6         BUF_PFN = .TEMP<9,15>;
: 1670 6         MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.I);
: 1671 6
: 1672 5         END;
: 1673 5
: 1674 5       CODECOMMENT

```

```

: 1675 5
: 1676 5
: 1677 5
: 1678 5
: 1679 5
: 1680 6
: 1681 6
: 1682 6
: 1683 6
: 1684 6
: L 1685 6
: xPRINT: Test 15, Subtest 0, Error 1
: L 1686 7
: xPRINT: Test 15, Subtest 0, Error 2
: 1687 6
: 1688 6
: 1689 6
: 1690 6
: 1691 7
: P 1692 7
: P 1693 7
: P 1694 7
: L 1695 7
: xPRINT: Test 15, Subtest 0, Error 3
: 1696 6
: 1697 6
: 1698 5
: 1699 5
: 1700 5
: 1701 5
: 1702 5
: 1703 4
: 1704 4
: 1705 3
: 1706 3
: 1707 2
: 1708 2
: 1709 1

```

```

''
'Execute the DATIP and after a nominal wait compare the contents',
'of the UET data register with the expected value. If',
'there is an error, report it.',
''
BEGIN
    UET_CTRL_REG = .TEMP3 OR DATIP;

    $DS_WAITUS(TIME=10);
    UBI_STATUS(.I,UBIMOD_DDP);      ! M7
    UET_STATUS(UBIMOD_DDP);        ! M7
    TEMP1 = .UET_DATA_REG;
    TEMP2 = .SCRATCH[0];
    IF .TEMP1 NEQ .TEMP2
    THEN
        BEGIN
            $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
                PRLINK = PRINT_GENERAL,
                P1 = FMT_BDP_DATI,P2 = 3,
                P3 = .I,P4 = .TEMP2,P5 = .TEMP1);
            Test 15, Subtest 0, Error 3
        END;
    END;
    TEMP = -1;
    MTPR(TEMP,x'37'); ! Cleanup the CMI      ! A8
END
WHILE $DS_INLOOP;      ! Scope loop ends here
END;
END;
$DS_ENDTEST;

```

```

.PSECT DISPATCH,NOWRT,NOEXE,2
00000000' 00000000' 00000000' 00000000' 00060 $: .ADDRESS P.AAM, P.AAN, P.AAO, TEST_015
00000000 00000003 00070 .LONG 3, 0
.PSECT $PLIT$,NOWRT,NOEXE,2
000F 00098 P.AAM: .WORD 15
20 61 74 61 44 20 64 65 72 65 66 66 75 42 1D 0009A P.AAN: .ASCII <29>\Buffered Data Path DATIP Test\
74 73 65 54 20 50 49 54 41 44 20 68 74 61 50 000A9
00000000 000B8 P.AAO: .LONG 0

```

[illegible]


```

                                7E D4 00067      CLRL  -(SP)
                                52 DD 00069      PUSHL BUF_PFN
                                53 DD 0006B      PUSHL MAP_NUMBER
                                04 FB 0006D      CALLS #4, MAP_SETUP
;
; Execute the DATIP and after a nominal wait compare the contents
; of the UET data register with the expected value. If
; there is an error, report it.
;
50      0000G CF 0003F464 8F C9 00072      BISL3 #259172, UNIBUS_SPACE, R0
60      57      03 A9 0007C      BISW3 #3, TEMP3, (R0)
                                7E      0A 7D 00080      MOVQ #10, -(SP)
                                00000000G 9F      02 FB 00083      CALLS #2, @DS$WAITUS
5A      59      02 78 0008A      ASHL #2, I, R10
50      5A      0000G CF C1 0008E      ADDL3 UBI_UNDER_TEST, R10, R0
0000' CF 60 1FFFFFFE 8F CB 00094      BICL3 #536870910, (R0), STATUS1
50      50      0000' CF D0 0009E      MOVL STATUS1, R0
                                1C 18 000A3      BGEQ 3$
                                7E 7C 000A5      CLRQ -(SP)
                                7E D4 000A7      CLRL -(SP)
50      DD 000A9      PUSHL R0
7E      D4 000AB      CLRL -(SP)
59      DD 000AD      PUSHL I
                                0000G CF 9F 000AF      PUSHAB PRINT_CSR_ERR
                                0000G CF 9F 000B3      PUSHAB UBIMOD_DDP
7E      0000G CF 9A 000B7      MOVZBL LUN, -(SP)
                                01 DD 000BC      PUSHL #1
50      6B      0A FB 000BE      CALLS #10, DS$ERRHARD
0000' CF 0003F464 8F C9 000C1 3$: BISL3 #259172, UNIBUS_SPACE, R0
0000' CF      60 B0 000CB      MOVW (R0), STATUS0
CF      FF1E 8F AA 000D0      BICW2 #65310, STATUS0
50      0000' CF 32 000D7      CVTWL STATUS0, R0
                                1D 13 000DC      BEQL 4$
                                7E 7C 000DE      CLRQ -(SP)
50      DD 000E0      PUSHL R0
7E      02 7D 000E2      MOVQ #2, -(SP)
                                0000G CF 9F 000E5      PUSHAB FMT_UET_STAT
                                0000G CF 9F 000E9      PUSHAB PRINT_GENERAL
                                0000G CF 9F 000ED      PUSHAB UBIMOD_DDP
7E      0000G CF 9A 000F1      MOVZBL LUN, -(SP)
                                02 DD 000F6      PUSHL #2
50      6B      0A FB 000F8      CALLS #10, DS$ERRHARD
0000G CF 0003F462 8F C9 000FB 4$: BISL3 #259170, UNIBUS_SPACE, R0
55      60 B0 00105      MOVW (R0), TEMP1
56      0000G CF B0 00108      MOVW SCRATCH, TEMP2
56      55 B1 0010D      CMPW TEMP1, TEMP2
                                22 13 00110      BEQL 5$
                                7E D4 00112      CLRL -(SP)
7E      55 3C 00114      MOVZWL TEMP1, -(SP)
7E      56 3C 00117      MOVZWL TEMP2, -(SP)
59      DD 0011A      PUSHL I
                                03 DD 0011C      PUSHL #3
                                0000G CF 9F 0011E      PUSHAB FMT_BDP_DATI
                                0000G CF 9F 00122      PUSHAB PRINT_GENERAL

```

ZZ-ECCBA-1.4 TEST_015: Buffered Data Path DATIP Test
UBI_TESTS_11_15
01-04 TEST_015: Buffered Data Path DATIP Test

L 8
6-Sep-1985 Fiche 2 Frame L8 Sequence 308
6-Sep-1985 17:19:55 VAX-11 Bliss-32 V4.1-746 Page 62
6-Sep-1985 17:14:57 [LEMIEUX.ECCBA.RELEASE]ECCBA5.B32;5 (7)

		0000G	CF	9F	00126	PUSHAB	UBIMOD_DDP	:
	7E	0000G	CF	9A	0012A	MOVZBL	LUN, -(SP)	:
			03	DD	0012F	PUSHL	#3	:
	6B		0A	FB	00131	CALLS	#10, DS\$ERRHARD	:
	54		01	CE	00134	MNEGL	#1, TEMP	: 1699
	37		54	DA	00137	MTPR	TEMP, #55	: 1700
	00000000G	9F	00	FB	0013A	CALLS	#0, a#DS\$INLOOP	: 1703
		03	50	E9	00141	BLBC	R0, 6\$	:
			FECE	31	00144	BRW	2\$	:
FEC4	59		03	F1	00147	ACBL	#3, #1, I, 1\$	: 1623
	00000000G	9F	00	FB	0014D	CALLS	#0, a#DS\$BREAK	: 1707
		50	01	D0	00154	MOVL	#1, R0	: 1709
				04	00157	RET		:

; Routine Size: 344 bytes, Routine Base: TEST_015 + 0000

ZZ-ECCBA-1.4 TEST_015: Buffered Data Path DATIP Test
UBI_TESTS_11_15
01-04 TEST_015: Buffered Data Path DATIP Test

M 8
6-Sep-1985 Fiche 2 Frame M8 Sequence 309
6-Sep-1985 17:19:55 VAX-11 Bliss-32 V4.1-746 Page 63
6-Sep-1985 17:14:57 [LEMIEUX.ECCBA.RELEASE]ECCBA5.B32;5 (8)

; 1710 1 \$DS_ENDMOD;
; 1711 1 END
; 1712 0 ELUDOM

.PSECT \$TSTCNT,NOWRT,NOEXE, OVR,2

0000000F 00000 L_TSTCNT:
.LONG 15 ;

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	26	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLIT\$	188	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
DISPATCH	120	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_011	720	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_012	305	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_013	1913	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_014	631	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_015	344	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$TSTCNT	4	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, OVR, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
SYSSCOMMON:[SYSLIB]STARLET.L32;3	10093	1	0	598	00:00.8
SYSSCOMMON:[SYSLIB]DIAG.L32;265	784	19	2	85	00:00.3

COMMAND QUALIFIERS

; BLISS/LIST ECCBA5.B32
; Size: 3913 code + 338 data bytes
; Run Time: 00:37.8
; Elapsed Time: 00:58.4
; Lines/CPU Min: 2718
; Lexemes/CPU-Min: 29369
; Memory Used: 481 pages

ZZ-ECCBA-1.4 TEST_015: Buffered Data Path DATIP Test
UBI TESTS_11_15
01-04 TEST_015: Buffered Data Path DATIP Test

; Compilation Complete

N 8

6-Sep-1985

Fiche 2 Frame N8

Sequence 310

6-Sep-1985 17:19:55

VAX-11 Bliss-32 V4.1-746

Page 64

```

; 0001 0  MODULE UBI_TESTS_16_22 ( ! UBI diagnostic program tests 16 to 22
; 0002 0      IDENT = '01-04'
; 0003 0      ) =
; 0004 1  BEGIN
; 0005 1
; 0006 1  !
; 0007 1  ! COPYRIGHT (C) 1980
; 0008 1  ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
; 0009 1  !
; 0010 1  ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
; 0011 1  ! COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
; 0012 1  ! ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
; 0013 1  ! MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
; 0014 1  ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
; 0015 1  ! TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
; 0016 1  ! REMAIN IN DEC.
; 0017 1  !
; 0018 1  ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
; 0019 1  ! AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
; 0020 1  ! CORPORATION.
; 0021 1  !
; 0022 1  ! DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
; 0023 1  ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
; 0024 1  !
; 0025 1  !
; 0026 1  ! **
; 0027 1  ! FACILITY:      VAX-11 Diagnostic Library
; 0028 1  !
; 0029 1  ! ABSTRACT:      This module contains tests 16 to 22 of the COMET
; 0030 1  !                  Unibus Interface Diagnostic Program.
; 0031 1  !
; 0032 1  ! ENVIRONMENT:    VAX-11 Diagnostic Supervisor
; 0033 1  !
; 0034 1  ! AUTHOR:         Rand Gray, CREATION DATE: 11-Dec-78
; 0035 1  !
; 0036 1  ! MODIFIED BY:    Don Monroe : May 1979
; 0037 1  !
; 0038 1  !             0-7      May 1979
; 0039 1  !                  Changed the format of the UBI MAP registers.
; 0040 1  !
; 0041 1  !             0-8      July 1979
; 0042 1  !                  Modified to support the real UET module.
; 0043 1  !
; 0044 1  !                  Fixed a BUG in XFER_SETUP routine. Loop that setup the maps
; 0045 1  !                  would not work if MAP_NUM was not zero.
; 0046 1  !
; 0047 1  !             0-9      July 1979
; 0048 1  !                  Changed referances to 'Byte Offset' to 'Byte Number'
; 0049 1  !
; 0050 1  !
; 0051 1  !             0-10     October 1979
; 0052 1  !                  Added a return value to the MAP_BUFFERS routine to support
; 0053 1  !                  the changes to Module 7 Revision 10.
; 0054 1  !                  Added byte-offset parameter to MAP_BUFFERS routine so routine
; 0055 1  !                  returns with one additional page per buffer.

```

```
: 0056 1 |
: 0057 1 |
: 0058 1 | 0-11 October 22,1979
: 0059 1 | Added a routine to print MAP errors. Called via PRINT LINK
: 0060 1 | in error hard calls.
: 0061 1 |
: 0062 1 | 0-12 December 7,1979
: 0063 1 | Fixed bug in PROBE_ADDRESS routine in the ADAWI builtin.
: 0064 1 |
: 0065 1 | 0-13 April 7,1980
: 0066 1 | Changed address of unibus space so second UBI will work.
: 0067 1 |
: 0068 1 | 0-14 April 8,1980
: 0069 1 | Added a print general routine and converted PRINTX after
: 0070 1 | error calls to print links.
: 0071 1 | 1-00 June 16,1980
: 0072 1 | Initial Release
: 0073 1 | 1-01 July 31,1980
: 0074 1 | Fixed bug in MAP_BUFFERS routine that would not detect
: 0075 1 | insufficient memory for the requested number of buffers.
: 0076 1 | 1-02 December 29,1980
: 0077 1 | Fixed bug in MAP_BUFFERS routine. Supervisor allocates memory
: 0078 1 | even if not enough for requested buffer size.
: 0079 1 | 1-03 May 6, 1981
: 0080 1 | Fixed bug in map_buffers routine.
: 0081 1 |
: 0082 1 | Kevin LeMieux
: 0083 1 |
: 0084 1 | 1-04 February,1985
: 0085 1 | Fixed bug in INIT code. If two DW's were specified; after the
: 0086 1 | first pass one of them one of them was no longer tested.
: 0087 1 | Fixed minor bugs in TEST 29.
: 0087 1 | --
```



```

: 0088 1 !
: 0089 1 ! TABLE OF CONTENTS:
: 0090 1 !
: 0091 1 !
: 0092 1 ! Buffered Data Path DATO Test
: 0093 1 ! Buffered Data Path DATOB Test
: 0094 1 ! Buffered Data Path Autopurge Test
: 0095 1 ! Byte Offset DATI Test
: 0096 1 ! Byte Offset DATIP/DATO Test
: 0097 1 ! Byte Offset DDP DATO Test
: 0098 1 ! Byte Offset BDP DATO Test
: 0099 1 !
: 0100 1 !
: 0101 1 !
: 0102 1 ! INCLUDE FILES:
: 0103 1 !
: 0104 1 !
: 0105 1 LIBRARY 'SYS$LIBRARY:STARLET';
: 0106 1 LIBRARY 'SYS$LIBRARY:DIAG';
: 0107 1 !
: 0108 1 !
: 0109 1 ! MACROS:
: 0110 1 !
: 0111 1 !
: 0112 1 !
: 0113 1 ! EQUATED SYMBOLS:
: 0114 1 !
: 0115 1 !
: 0116 1 LITERAL
: 0117 1 ALL_ONES = %X'FFFFFFFF',
: 0118 1 ALL_ZEROS = 0,
: 0119 1 BR4 = 3,
: 0120 1 BR5 = 5,
: 0121 1 BR6 = 9,
: 0122 1 BR7 = 17,
: 0123 1 CSR_MASK = %X'E0000001',
: 0124 1
: 0125 1 DATI = 1,
: 0126 1 DATIP = 3,
: 0127 1 DATO = 5,
: 0128 1 DATOB = 7,
: 0129 1
: 0130 1 DIAG_CHK_MODE = %X'C000000',
: 0131 1 DISABLE_ECC = %X'2000000',
: 0132 1 ENABLE_ECC = %X'2000000',
: 0133 1 NORMAL = %X'E000000',
: 0134 1 ERR = BIT31,
: 0135 1 MAP_MASK = %X'82607FFF',
: 0136 1 MEMORY_CNT = %X'F20000',
: 0137 1 NON_XIST_NEXUS = %X'F40000',
: 0138 1 NXM = BIT30,
: 0139 1 PURGE = BIT0,
: 0140 1 UCE = BIT29;
: 0141 1 !
: 0142 1 ! OWN STORAGE:

```

! M7

```

: 0143 1 !
: 0144 1 ! OWN
: 0145 1 !
: 0146 1 ! BYTE_PATTERN: VECTOR[4,BYTE], ! Data patterns loaded at run time
: 0147 1 ! STATUS0: SIGNED WORD, ! Used by the UET status macro
: 0148 1 ! STATUS1, ! Used by the UBI status macro
: 0149 1 ! WORD_PATTERN: VECTOR[5,WORD]; ! Data patterns loaded at run time
: 0150 1 !
: 0151 1 ! EXTERNAL REFERENCES:
: 0152 1 !
: 0153 1 EXTERNAL ROUTINE
: 0154 1 ! DECODER,
: 0155 1 ! ENCODER,
: 0156 1 ! MAP_SETUP,
: 0157 1 ! PRINT_GENERAL:NOVALUE,
: 0158 1 ! PROBE_ADDRESS,
: 0159 1 ! UNIBUS_SIZER,
: 0160 1 ! XFER_SETUP;
: 0161 1
: 0162 1 EXTERNAL
: 0163 1 ! BUFFER_SETUP,
: 0164 1 ! CODEWORDS: VECTOR[9,WORD],
: 0165 1 ! UBI_UNDER_TEST,
: 0166 1 ! UNIBUS_SPACE,
: 0167 1 ! DECODED_WORDS: VECTOR[9],
: 0168 1 ! EVENT_FLAG: BITVECTOR[4],
: 0169 1 ! FMT_BDP-DAT1,
: 0170 1 ! FMT_BDP-DATO,
: 0171 1 ! FMT_BDP-DATOB,
: 0172 1 ! FMT_BYTE_OFF,
: 0173 1 ! FMT_CMI_UB,
: 0174 1 ! FMT_CPU_RW,
: 0175 1 ! FMT-DDP-DAT1,
: 0176 1 ! FMT-DDP-DATO,
: 0177 1 ! FMT-DDP-DATOB,
: 0178 1 ! FMT-DP_SEL,
: 0179 1 ! FMT_MAP_ADDR,
: 0180 1 ! FMT_MAP_BIT, ! D13
: 0181 1 ! FMT_MAP_CS,
: 0182 1 ! FMT_MAP-DB-SA0, ! M8
: 0183 1 ! FMT_MAP-DB-SA1, ! A8
: 0184 1 ! FMT_MAP_FAIL,
: 0185 1 ! FMT-MCHK,
: 0186 1 ! FMT-UA1,
: 0187 1 ! FMT-UB_CMI,
: 0188 1 ! FMT-UET_STAT,
: 0189 1 ! FREE_MAP: VECTOR[4,WORD],
: 0190 1 ! HANDLER,
: 0191 1 ! INTERRUPT_EXP: BYTE,
: 0192 1 ! INTERRUPT_OCC: BYTE,
: 0193 1 ! LUN: BYTE,
: 0194 1 ! MAP_BUFFERS,
: 0195 1 ! NOISY_WORDS: VECTOR[9,WORD],
: 0196 1 ! NUM_UB,
: 0197 1 ! PRINT_CSR_ERR,

```

```

; 0198 1 PRINT_DCOMP_ERR,
; 0199 1 UBIMOD_BDP,
; 0200 1 UBIMOD_BYTE_OFF,
; 0201 1 UBIMOD_CMI,
; 0202 1 UBIMOD_CPU_RW,
; 0203 1 UBIMOD_CSR,
; 0204 1 UBIMOD_DDP,
; 0205 1 UBIMOD_DP_SEL,
; 0206 1 UBIMOD_MAP,
; 0207 1 UBIMOD_MAP_CS,
; 0208 1 UBIMOD_MAP_ADDR,
; 0209 1 UBIMOD_MAP_CTRL,
; 0210 1 UBIMOD_UA1,
; 0211 1 UBIMOD_UB_CMI,
; 0212 1 SCRATCH: VECTOR[128],
; 0213 1 UBE_BASE_ADDR: VECTOR[4],
; 0214 1 UBE_BUF_SIZE,
; 0215 1 ! UBE_ERROR, ! D8
; 0216 1 UBE_STAT_ERR: BITVECTOR[4],
; 0217 1 UBE0_ISR,
; 0218 1 UET_ISR,
; 0219 1 VALID_MAPS: BITVECTOR[512];
; 0220 1
; 0221 1 MACRO
; 0222 1 !++
; 0223 1 ! This macro defines the address of the control and status register
; 0224 1 ! for the given buffered data path number.
; 0225 1 !--
; 0226 1 UBI_CSR(BDP) = (.UBI_UNDER_TEST + BDP*4)x,
; 0227 1
; 0228 1 !++
; 0229 1 ! This macro defines the address of a UBI map for the given map
; 0230 1 ! number.
; 0231 1 !--
; 0232 1 UBI_MAP(NUM) = (.UBI_UNDER_TEST + %X'800' + NUM*4)x;
; 0233 1
; 0234 1 !++
; 0235 1 ! These macros temporarily define the addresses of the (simulated) UET
; 0236 1 ! registers. They will be replaced for the real UET.
; 0237 1 !--
; 0238 1 MACRO ! A8
; 0239 1 UET_ADDR_REG = (.UNIBUS_SPACE OR %X'3F460')<0,16>x, ! A8 M13
; 0240 1 UET_DATA_REG = (.UNIBUS_SPACE OR %X'3F462')<0,16>x, ! A8 M13
; 0241 1 UET_DATA_REGB = (.UNIBUS_SPACE OR %X'3F462')<0,8>x, ! A8 M13
; 0242 1 UET_CTRL_REG = (.UNIBUS_SPACE OR %X'3F464')<0,16>x, ! A8 M13
; 0243 1
; 0244 1 !++ ! A8
; 0245 1 ! This macro is used for checking the status of the UET. ! A8
; 0246 1 !-- ! A8
; M 0247 1 UET_STATUS(CALLOUT) = ! A8
; M 0248 1 STATUS0 = .UET_CTRL_REG; ! Read the UET control register ! A8
; M 0249 1 STATUS0 = .STATUS0 AND %X'E1'; ! A8
; M 0250 1 IF .STATUS0 NEQ 0 ! A8
; M 0251 1 THEN ! A8
; M 0252 1 BEGIN ! A8

```



```

; M 0253 1          $DS_ERRHARD(UNIT=.LUN,MSGADR=CALLOUT,
; M 0254 1          PRLINK = PRINT_GENERAL,
; M 0255 1          P1 = FMT_UET_STAT,P2 = 2,
; M 0256 1          P3 = 0,P4 = .STATUS0);          ! A8
; 0257 1          END;x;                          ! A8
; 0258 1          MACRO                          ! A8
; 0259 1          !++
; 0260 1          ! This macro is used for checking the status of the UBI.
; 0261 1          !--
; M 0262 1          UBI_STATUS(BDP,CALLOUT) =
; M 0263 1          STATUS1 = .UBI_CSR(BDP) AND CSR_MASK;
; M 0264 1          IF .STATUS1 LSS 0
; M 0265 1          THEN
; M 0266 1          $DS_ERRHARD(UNIT=.LUN,
; M 0267 1          MSGADR=CALLOUT,
; M 0268 1          PRLINK=PRINT_CSR_ERR,
; 0269 1          P1=BDP,P2=0,P3=.STATUS1);x;
; 0270 1
; 0271 1          !
; 0272 1          ! Required Diagnostic Supervisor macros.
; 0273 1          !
; L 0274 1          $DS_BGNMOD(ENV=0,TEST=16);
; xPRINT:          Bliss-32 Diagnostic Macro Library version 7.0 "DIAG.L32 (57)"
; 0275 1          $DS_DSADEF;
; 0276 1          $DS_SECDEF(ALL,UBE);
; 0277 1
; 0278 1          BUILTIN
; 0279 1          ROT;

```

! This is so we can do ROTLs

TEST_016: Buffered Data Path DATO Test

\$DS_BGNTTEST (SECTION=(DEFAULT,ALL),TEXT='Buffered Data Path DATO Test');

!++
! TEST DESCRIPTION:

This test verifies the ability of the UBI to handle DATO transactions through each of the buffered data paths (BDPs). The UBI behavior for a DATO is primarily dependent on the contents of the buffer. The first case is when the buffer has Unibus data, no address match. The second case is when the buffer is empty or contains CMI data. The third case is when the buffer has Unibus data, and there is an address match.

The test algorithm is as follows. Purge the BDP, clear two CMI locations, load the UET data register with a unique pattern, then execute a DATO. Change the data in the UET data register, execute another DATO with a different destination address (it must be outside the original longword). After the first DATO, check that the CMI destination is still at its initialized value. After the second DATO, check that the CMI destination contains the first unique pattern. Execute a third DATO, changing Unibus address bit 1, and also changing the data in the UET data register, and check that the second CMI location now contains the second and third data patterns, concatenated into a longword. This effectively tests the correct functionality of the byte flags.

ASSUMPTIONS:

Test 1-Test 15

REGISTER

BUF_PFN,
M,
MAP_NUMBER,
N,
TEMP,
TEMP3,
UBUS_ADDR: WORD;

MACRO

UBUS_PF = UBUS_ADDR<9,7>X, ! Defines the Unibus page frame field
UBUS_BO = UBUS_ADDR<0,9>X; ! Defines the Unibus byte number field

CODECOMMENT

'Call the Unibus sizer routine to locate any Unibus memory. This builds',
'a table of useable map entries. In the test which follows, the map',
'selected is drawn from this table.'

BEGIN

UNIBUS_SIZER();

END;

```

: 0335 2
: 0336 2 CODECOMMENT
: 0337 2 ''
: 0338 2 'Clear two successive longword aligned CMI locations. Put a unique data',
: 0339 2 'pattern into the UET data register. Set up the UBI for a DATO',
: 0340 2 'addressing the first CMI longword, execute it, and check that both CMI',
: 0341 2 'destinations still contain zero. Change the UET address',
: 0342 2 'register to point to the second CMI longword, change the data pattern',
: 0343 2 'in the UET data pattern, and execute the second DATO. After a',
: 0344 2 'nominal wait, check that the first unique data pattern is now in the',
: 0345 2 'first CMI destination, and then check that the second CMI destination',
: 0346 2 'is still empty. Load the UET data register with a third unique',
: 0347 2 'pattern, change the UET address register bit 1 from a zero to a',
: 0348 2 'one, and execute a third DATO. The second CMI destination should now',
: 0349 2 'contain the second and third unique data patterns. This checks the',
: 0350 2 'correct functionality of the byte flags.',
: 0351 2 ''
: 0352 3 BEGIN
: 0353 3
: 0354 3 INCR N FROM 1 TO 3 DO
: 0355 4 BEGIN
: 0356 4
: 0357 4 DO ! Scope loop begins here
: 0358 5 BEGIN
: 0359 5
: 0360 5 UBI_CSR(.N) = PURGE;
: 0361 5 $DS_WAITUS(TIME=10); ! Wait for the purge to happen
: 0362 5 SCRATCH[0] = 0;
: 0363 5 SCRATCH[1] = 0;
: 0364 5
: 0365 5 CODECOMMENT
: 0366 5 ''
: 0367 5 'Set up the UET address register with the Unibus',
: 0368 5 'address. Bits 9 to 15 (the Unibus page frame) contain',
: 0369 5 'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
: 0370 5 'the byte number field of the CMI address.',
: 0371 5 ''
: 0372 6 BEGIN
: 0373 6
: 0374 6 UET_DATA_REG = %X'1357';
: 0375 6 MAP_NUMBER = .FREE_MAP[.N];
: 0376 6 UBUS_PF = .MAP_NUMBER;
: 0377 6 UBUS_BO = SCRATCH[0];
: 0378 6 UET_ADDR_REG = .UBUS_ADDR;
: 0379 6
: 0380 5 END;
: 0381 5
: 0382 5 CODECOMMENT
: 0383 5 ''
: 0384 5 'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
: 0385 5 'map number.',
: 0386 5 ''
: 0387 6 BEGIN
: 0388 6
: 0389 6 TEMP3 = .MAP_NUMBER<7,2> * 8;

```



```

: 0390 6
: 0391 5      END;
: 0392 5
: 0393 5      CODECOMMENT
: 0394 5      ''
: 0395 5      'Set up the UBI map. Put the CMI page frame number (PFN) into',
: 0396 5      'the register BUF_PFN. Then call the map setup routine with the',
: 0397 5      'parameters map number, PFN of the CMI buffer, no address',
: 0398 5      'offset, and data path number.',
: 0399 5      ''
: 0400 6      BEGIN
: 0401 6
: 0402 6      TEMP = SCRATCH[0];
: 0403 6      BUF_PFN = .TEMP<9,15>;
: 0404 6      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.N);
: 0405 6
: 0406 5      END;
: 0407 5
: 0408 5      CODECOMMENT
: 0409 5      ''
: 0410 5      'Execute the first DATO and after a nominal wait check that the',
: 0411 5      'CMI destinations both still contain zero.',
: 0412 5      ''
: 0413 6      BEGIN
: 0414 6
: 0415 6      UET_CTRL_REG = .TEMP3 OR DATO;
: 0416 6
: 0417 6      $DS_WAITUS(TIME=10);
: L 0418 6      UBI_STATUS(.N,UBIMOD_BDP);          ! M7
: %PRINT:      Test 16, Subtest 0, Error 1
: L 0419 7      UET_STATUS(UBIMOD_BDP);          ! M7
: %PRINT:      Test 16, Subtest 0, Error 2
: 0420 6      IF .SCRATCH[0] NEQ 0
: 0421 6      THEN
: 0422 7      BEGIN
: P 0423 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
: P 0424 7      PRLINK = PRINT_GENERAL,
: P 0425 7      P1 = FMT_BDP_DATO,P2 = 3,
: L 0426 7      P3 = .N,P4 = 0,P5 = .SCRATCH[0]);
: %PRINT:      Test 16, Subtest 0, Error 3
: 0427 6      END;
: 0428 6      IF .SCRATCH[1] NEQ 0
: 0429 6      THEN
: 0430 7      BEGIN
: P 0431 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
: P 0432 7      PRLINK = PRINT_GENERAL,
: P 0433 7      P1 = FMT_BDP_DATO,P2 = 3,
: L 0434 7      P3 = .N,P4 = 0,P5 = .SCRATCH[1]);
: %PRINT:      Test 16, Subtest 0, Error 4
: 0435 6      END;
: 0436 6
: 0437 5      END;
: 0438 5
: 0439 5      CODECOMMENT
: 0440 5      ''

```

```

; 0441 5      'Set up the UET address register with the Unibus',
; 0442 5      'address. Bits 9 to 15 (the Unibus page frame) contain',
; 0443 5      'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
; 0444 5      'the byte number field of the CMI address.',
; 0445 5      '':
; 0446 6      BEGIN
; 0447 6
; 0448 6      UET_DATA_REG = %X'5678';
; 0449 6      UBUS_PF = .MAP_NUMBER;
; 0450 6      UBUS_BO = SCRATCH[1];
; 0451 6      UET_ADDR_REG = .UBUS_ADDR;
; 0452 6
; 0453 5      END;
; 0454 5
; 0455 5      CODECOMMENT
; 0456 5      '':
; 0457 5      'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
; 0458 5      'map number.',
; 0459 5      '':
; 0460 6      BEGIN
; 0461 6
; 0462 6      TEMP3 = .MAP_NUMBER<7,2> * 8;
; 0463 6
; 0464 5      END;
; 0465 5
; 0466 5      CODECOMMENT
; 0467 5      '':
; 0468 5      'Set up the UBI map. Put the CMI page frame number (PFN) into',
; 0469 5      'the register BUF_PFN. Then call the map setup routine with the',
; 0470 5      'parameters map number, PFN of the CMI buffer, no address',
; 0471 5      'offset, and data path number.',
; 0472 5      '':
; 0473 6      BEGIN
; 0474 6
; 0475 6      TEMP = SCRATCH[1];
; 0476 6      BUF_PFN = .TEMP<9,15>;
; 0477 6      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.N);
; 0478 6
; 0479 5      END;
; 0480 5
; 0481 5      CODECOMMENT
; 0482 5      '':
; 0483 5      'Execute the second DATO and after a nominal wait check that the',
; 0484 5      'first CMI destination contains the expected data pattern.',
; 0485 5      'Also, check that the second CMI destination is still zero.',
; 0486 5      '':
; 0487 6      BEGIN
; 0488 6
; 0489 6      UET_CTRL_REG = .TEMP3 OR DATO;
; 0490 6
; 0491 6      $DS_WAITUS(TIME=10);
; L 0492 6      UBI_STATUS(.N,UBIMOD_BDP);          ! M7
; %PRINT:      Test 16, Subtest 0, Error 5
; L 0493 7      UET_STATUS(UBIMOD_BDP);          ! M7
; %PRINT:      Test 16, Subtest 0, Error 6

```

```

: 0494 6      IF .SCRATCH[0] NEQ %X'1357'
: 0495 6      THEN
: 0496 7      BEGIN
: P 0497 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
: P 0498 7      PRLINK = PRINT_GENERAL,
: P 0499 7      P1 = FMT_BDP_DATO,P2 = 3,
: L 0500 7      P3 = .N,P4 = %X'1357',P5 = .SCRATCH[0]);
: %PRINT:      Test 16, Subtest 0, Error 7
: 0501 6      END;
: 0502 6      IF .SCRATCH[1] NEQ 0
: 0503 6      THEN
: 0504 7      BEGIN
: P 0505 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
: P 0506 7      PRLINK = PRINT_GENERAL,
: P 0507 7      P1 = FMT_BDP_DATO,P2 = 3,
: L 0508 7      P3 = .N,P4 = 0,P5 = .SCRATCH[1]);
: %PRINT:      Test 16, Subtest 0, Error 8
: 0509 6      END;
: 0510 6
: 0511 5      END;
: 0512 5
: 0513 5      CODECOMMENT
: 0514 5      ''
: 0515 5      'Change the data in the UET data register and change',
: 0516 5      'Unibus address bit 1 in the UET address register from',
: 0517 5      'zero to one.',
: 0518 5      '';
: 0519 6      BEGIN
: 0520 6
: 0521 6      UET_DATA_REG = %X'1234';
: 0522 6      UET_ADDR_REG = .UBUS_ADDR + 2;
: 0523 6
: 0524 5      END;
: 0525 5
: 0526 5      CODECOMMENT
: 0527 5      ''
: 0528 5      'Execute the third DATO and after a nominal wait check that the',
: 0529 5      'second CMI destination now contains the expected data pattern.',
: 0530 5      '';
: 0531 6      BEGIN
: 0532 6
: 0533 6      UET_CTRL_REG = .TEMP3 OR DATO;
: 0534 6
: 0535 6      $DS_WAITUS(TIME=10);
: L 0536 6      UBI_STATUS(.N,UBIMOD_BDP);      ! M7
: %PRINT:      Test 16, Subtest 0, Error 9
: L 0537 7      UET_STATUS(UBIMOD_BDP);      ! M7
: %PRINT:      Test 16, Subtest 0, Error 10
: 0538 6      IF .SCRATCH[1] NEQ %X'12345678'
: 0539 6      THEN
: 0540 7      BEGIN
: P 0541 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
: P 0542 7      PRLINK = PRINT_GENERAL,
: P 0543 7      P1 = FMT_BDP_DATO,P2 = 3,
: L 0544 7      P3 = .N,P4 = %X'12345678',P5 = .SCRATCH[1]);

```


ZZ-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_016: Buffered Data Path DATO Test
TEST_016: Buffered Data Path DATO Test

M 9

6-Sep-1985

Fiche 2

Frame M9

Sequence 322

6-Sep-1985 17:20:57

VAX-11 Bliss-32 V4.1-746

Page 12

6-Sep-1985 17:15:01

[LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5

(3)

```
; XPRINT:
; 0545 6
; 0546 6
; 0547 5
; 0548 5
; 0549 5
; 0550 4
; 0551 4
; 0552 3
; 0553 3
; 0554 2
; 0555 2
; 0556 1

Test 16, Subtest 0, Error 11
END;
END;
END
WHILE $DS_INLOOP;
END;
END;
$DS_ENDTEST;
```

.TITLE UBI_TESTS_16_22
.IDENT \01-04\

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00000 \$:
00000000 00000003 00010

.ADDRESS P.AAA, P.AAB, P.AAC, TEST_016
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

```
20 61 74 61 44 20 64 65 72 65 66 66 75 42 1C 0010 00000 P.AAA: .WORD 16
74 73 65 54 20 4F 54 41 44 20 68 74 61 50 00002 P.AAB: .ASCII <28>\Buffered Data Path DATO Test\
00011
0001F .BLKB 1
00020 P.AAC: .LONG 0
```

.PSECT \$OWN\$,NOEXE,2

00000 BYTE_PATTERN:
.BLKB 4
00004 STATUS0: .BLKB 2
00006 .BLKB 2
00008 STATUS1: .BLKB 4
0000C WORD_PATTERN:
.BLKB 10

DSA\$AL_APTMAIL= 65024
DSA\$GL_FLAGS= 65024
DSA\$GL_APTCOM= 65028
DSA\$GL_PASSES= 65032
DSA\$GL_UNITS= 65036
DSA\$GL_SECTNO= 65040
DSA\$GL_SID= 65044
DSA\$GL_MSGTYP= 65088
DSA\$GL_ERRNO= 65092
DSA\$GL_EVENT= 65096
DSA\$GL_SUBTNO= 65100
DSA\$GL_TESTNO= 65104
DSA\$GL_PASSNO= 65108
DSA\$GL_DEVLEN= 65112

ZZ-ECCBA-1.4 TEST_016: Buffered Data Path DATO Test
UBI_TESTS_16_22
01-04 TEST_016: Buffered Data Path DATO Test

N 9
6-Sep-1985 Fiche 2 Frame N9 Sequence 323
6-Sep-1985 17:20:57 VAX-11 Bliss-32 V4.1-746 Page 13
6-Sep-1985 17:15:01 [LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5 (3)

```
DSA$GL_DEVNAM= 65116
DSA$GL_MSGPTR= 65128
.EXTRN DECODER, ENCODER
.EXTRN MAP_SETUP, PRINT_GENERAL
.EXTRN PROBE_ADDRESS, UNIBUS_SIZER
.EXTRN XFER_SETUP, BUFFER_SETUP
.EXTRN CODEWORDS, UBI_UNDER_TEST
.EXTRN UNIBUS_SPACE, DECODED_WORDS
.EXTRN EVENT_FLAG, FMT_BDP_DATI
.EXTRN FMT_BDP_DATO, FMT_BDP_DATO
.EXTRN FMT_BYTE_OFF, FMT_CMI_UB
.EXTRN FMT_CPU_RW, FMT_DDP_DATI
.EXTRN FMT_DDP_DATO, FMT_DDP_DATO
.EXTRN FMT_DP_SEL, FMT_MAP_ADDR
.EXTRN FMT_MAP_CS, FMT_MAP_DB_SA0
.EXTRN FMT_MAP_DB_SA1, FMT_MAP_FAIL
.EXTRN FMT_MCHK, FMT_UA1
.EXTRN FMT_UB_CMI, FMT_UET_STAT
.EXTRN FREE_MAP, HANDLER
.EXTRN INTERRUPT_EXP, INTERRUPT_OCC
.EXTRN LUN, MAP_BUFFERS
.EXTRN NOISY_WORDS, NUM_UBE
.EXTRN PRINT_CSR_ERR, PRINT_DCOMP_ERR
.EXTRN UBIMOD_BDP, UBIMOD_BYTE_OFF
.EXTRN UBIMOD_CMI, UBIMOD_CPU_RW
.EXTRN UBIMOD_CSR, UBIMOD_DDP
.EXTRN UBIMOD_DP_SEL, UBIMOD_MAP
.EXTRN UBIMOD_MAP_CS, UBIMOD_MAP_ADDR
.EXTRN UBIMOD_MAP_CTRL
.EXTRN UBIMOD_UA1, UBIMOD_UB_CMI
.EXTRN SCRATCH, UBE_BASE_ADDR
.EXTRN UBE_BUF_SIZE, UBE_STAT_ERR
.EXTRN UBE0_ISR, UET_ISR
.EXTRN VALID_MAPS, DS$WAITUS
.EXTRN DS$ERRHARD, DS$INLOOP
.EXTRN DS$BREAK

.PSECT TEST_016,NOWRT,2

.ENTRY TEST_016, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0280
R11;
MOVAB SCRATCH, R11;
MOVAB STATUS0, R10;
MOVAB a#DS$ERRHARD, R9;

; Call the Unibus sizer routine to locate any Unibus memory. T
; his builds
; a table of useable map entries. In the test which follows, t
; he map
; selected is drawn from this table.
;
0000G CF 00 FB 00013 CALLS #0, UNIBUS_SIZER; 0332
; Clear two successive longword aligned CMI locations. Put a u
; nique data
```

```
; pattern into the UET data register. Set up the UBI for a DAT
;
; addressing the first CMI longword, execute it, and check that
; both CMI
; destinations still contain zero. Change the UET address
; register to point to the second CMI longword, change the data
; pattern
; in the UET data pattern, and execute the second DATO. After
; a
; nominal wait, check that the first unique data pattern is now
; in the
; first CMI destination, and then check that the second CMI des
; tination
; is still empty. Load the UET data register with a third uniq
; ue
; pattern, change the UET address register bit 1 from a zero to
; a
; one, and execute a third DATO. The second CMI destination sh
; ould now
; contain the second and third unique data patterns. This chec
; ks the
; correct functionality of the byte flags.
```

```
58      57      01 D0 00018
50      57      02 78 0001B
      58      CF C1 0001F
      60      01 D0 00025
      7E      0A 7D 00028
00000000G 9F      02 FB 0002B
      6B      7C 00032
```

```
1$:      MOVL      #1, N ; 0354
2$:      ASHL      #2, N, R8 ; 0360
      ADDL3      UBI_UNDER_TEST, R8, R0 ;
      MOVL      #1, (R0) ;
      MOVQ      #10, -(SP) ; 0361
      CALLS      #2, a#DS$WAITUS ;
      CLRQ      SCRATCH ; 0362
```

```
; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.
```

```
50      0000G CF 0003F462 8F C9 00034
      60      1357 8F B0 0003E
56      07      53      0000GCF47 3C 00043
      09      53 F0 00049
      50      6B 9E 0004E
56      09      00 F0 00051
      50      0000G CF 0003F460 8F C9 00056
      60      56 B0 00060
```

```
      BISL3      #259170, UNIBUS_SPACE, R0 ; 0372
      MOVW      #4951, (R0) ; 0374
      MOVZWL     FREE_MAP[N], MAP_NUMBER ; 0375
      INSV      MAP_NUMBER, #9, #7, UBUS_ADDR ; 0376
      MOVAB      SCRATCH, R0 ; 0377
      INSV      R0, #0, #9, UBUS_ADDR ;
      BISL3      #259168, UNIBUS_SPACE, R0 ;
      MOVW      UBUS_ADDR, (R0) ; 0378
```

```
; Write the 2 MSBs of the Unibus address with the 2 MSBs of the
; map number.
```

```
55      53      02      07 EF 00063
      55      08 C4 00068
```

```
      EXTZV      #7, #2, MAP_NUMBER, TEMP3 ; 0389
      MULL2      #8, TEMP3 ;
```

```
; Set up the UBI map. Put the CMI page frame number (PFN) into
; the register BUF_PFN. Then call the map setup routine with t
; he
; parameters map number, PFN of the CMI buffer, no address
```


; offset, and data path number.

52	54	54	6B	9E	0006B	MOVAB	SCRATCH, TEMP	; 0402
		0F	09	EF	0006E	EXTZV	#9, #15, TEMP, BUF_PFN	; 0403
			57	DD	00073	PUSHL	N	; 0404
			7E	D4	00075	CLRL	-(SP)	:
			52	DD	00077	PUSHL	BUF_PFN	:
			53	DD	00079	PUSHL	MAP_NUMBER	:
	0000G	CF	04	FB	0007B	CALLS	#4, MAP_SETUP	:

; Execute the first DATO and after a nominal wait check that the
; CMI destinations both still contain zero.

50	0000G	CF	0003F464	8F	C9	00080	BISL3	#259172, UNIBUS_SPACE, R0	; 0413
60		55		05	A9	0008A	BISW3	#5, TEMP3, (R0)	; 0415
		7E		0A	7D	0008E	MOVQ	#10, -(SP)	; 0417
	00000000G	9F		02	FB	00091	CALLS	#2, @DS\$WAITUS	:
04	AA	0000GDF47	1FFFFFFE	8F	CB	00098	BICL3	#536870910, @UBI_UNDER_TEST[N], STATUS1	; 0418
		50	04	AA	D0	000A4	MOVL	STATUS1, R0	:
				1C	18	000A8	BGEQ	3\$	:
				7E	7C	000AA	CLRQ	-(SP)	:
				7E	D4	000AC	CLRL	-(SP)	:
				50	DD	000AE	PUSHL	R0	:
				7E	D4	000B0	CLRL	-(SP)	:
				57	DD	000B2	PUSHL	N	:
		0000G		CF	9F	000B4	PUSHAB	PRINT_CSR_ERR	:
		0000G		CF	9F	000B8	PUSHAB	UBIMOD_BDP	:
	7E	0000G		CF	9A	000BC	MOVZBL	LUN, -(SP)	:
				01	DD	000C1	PUSHL	#1	:
				0A	FB	000C3	CALLS	#10, DS\$ERRHARD	:
50	0000G	CF	0003F464	8F	C9	000C6	BISL3	#259172, UNIBUS_SPACE, R0	; 0419
		6A		60	B0	000D0	MOVW	(R0), STATUS0	:
		6A	FF1E	8F	AA	000D3	BICW2	#65310, STATUS0	:
		50		6A	32	000D8	CVTWL	STATUS0, R0	:
				1D	13	000DB	BEQL	4\$	:
				7E	7C	000DD	CLRQ	-(SP)	:
				50	DD	000DF	PUSHL	R0	:
	7E			02	7D	000E1	MOVQ	#2, -(SP)	:
		0000G		CF	9F	000E4	PUSHAB	FMT_UET_STAT	:
		0000G		CF	9F	000E8	PUSHAB	PRINT_GENERAL	:
		0000G		CF	9F	000EC	PUSHAB	UBIMOD_BDP	:
	7E	0000G		CF	9A	000F0	MOVZBL	LUN, -(SP)	:
				02	DD	000F5	PUSHL	#2	:
				0A	FB	000F7	CALLS	#10, DS\$ERRHARD	:
69				6B	D0	000FA	MOVL	SCRATCH, R0	; 0420
50				20	13	000FD	BEQL	5\$	:
				7E	D4	000FF	CLRL	-(SP)	; 0426
				50	DD	00101	PUSHL	R0	:
				7E	D4	00103	CLRL	-(SP)	:
				57	DD	00105	PUSHL	N	:
				03	DD	00107	PUSHL	#3	:
		0000G		CF	9F	00109	PUSHAB	FMT_BDP_DATO	:
		0000G		CF	9F	0010D	PUSHAB	PRINT_GENERAL	:
		0000G		CF	9F	00111	PUSHAB	UBIMOD_BDP	:

```

7E      0000G  CF  9A 00115      MOVZBL  LUN, -(SP)
03      DD 0011A      PUSHL  #3
69      0A  FB 0011C      CALLS  #10, DS$ERRHARD
50      04  AB  D0 0011F  5$:  MOVL  SCRATCH+4, R0
20      13  00123      BEQL  6$
7E      D4  00125      CLRL  -(SP)
50      DD 00127      PUSHL  R0
7E      D4  00129      CLRL  -(SP)
57      DD 0012B      PUSHL  N
03      DD 0012D      PUSHL  #3
0000G   CF  9F 0012F      PUSHAB  FMT_BDP_DATO
0000G   CF  9F 00133      PUSHAB  PRINT_GENERAL
0000G   CF  9F 00137      PUSHAB  UBIMOD_BDP
7E      0000G  CF  9A 0013B      MOVZBL  LUN, -(SP)
04      DD 00140      PUSHL  #4
69      0A  FB 00142      CALLS  #10, DS$ERRHARD

; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.
50      0000G  CF  0003F462  8F  C9 00145  6$:  BISL3  #259170, UNIBUS_SPACE, R0
60      5678      8F  B0 0014F      MOVW  #22136, (R0)
56      07      09      53  F0 00154      INSV  MAP_NUMBER, #9, #7, UBUS_ADDR
50      04      AB  9E 00159      MOVAB  SCRATCH+4, R0
56      09      00      50  F0 0015D      INSV  R0, #0, #9, UBUS_ADDR
50      0000G  CF  0003F460  8F  C9 00162      BISL3  #259168, UNIBUS_SPACE, R0
60      56      56  B0 0016C      MOVW  UBUS_ADDR, (R0)

; Write the 2 MSBs of the Unibus address with the 2 MSBs of the
; map number.
55      53      02      07  EF 0016F      EXTZV  #7, #2, MAP_NUMBER, TEMP3
55      55      08  C4 00174      MULL2  #8, TEMP3

; Set up the UBI map. Put the CMI page frame number (PFN) into
; the register BUF_PFN. Then call the map setup routine with t
; he
; parameters map number, PFN of the CMI buffer, no address
; offset, and data path number.
52      54      54      04  AB  9E 00177      MOVAB  SCRATCH+4, TEMP
0F      09  EF 0017B      EXTZV  #9, #15, TEMP, BUF_PFN
57      DD 00180      PUSHL  N
7E      D4  00182      CLRL  -(SP)
52      DD 00184      PUSHL  BUF_PFN
53      DD 00186      PUSHL  MAP_NUMBER
0000G   CF  04  FB 00188      CALLS  #4, MAP_SETUP

; Execute the second DATO and after a nominal wait check that t
; he
; first CMI destination contains the expected data pattern.
; Also, check that the second CMI destination is still zero.

```

ZZ-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_016: Buffered Data Path DATO Test
TEST_016: Buffered Data Path DATO Test

E 10

6-Sep-1985

Fiche 2 Frame E10

Sequence 327

6-Sep-1985 17:20:57

VAX-11 Bliss-32 V4.1-746

Page 17

6-Sep-1985 17:15:01

[LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5

(3)

50	0000G	CF	0003F464	8F	C9	0018D	BISL3	#259172, UNIBUS_SPACE, R0	:	0487
60		55		05	A9	00197	BISW3	#5, TEMP3, (R0)	:	0489
		7E		0A	7D	0019B	MOVQ	#10, -(SP)	:	0491
	00000000G	9F		02	FB	0019E	CALLS	#2, a#DS\$WAITUS	:	
04	AA	0000GDF47	1FFFFFFE	8F	CB	001A5	BICL3	#536870910, aUBI_UNDER_TEST[N], STATUS1	:	0492
		50	04	AA	D0	001B1	MOVL	STATUS1, R0	:	
				1C	18	001B5	BGEQ	7\$	:	
				7E	7C	001B7	CLRQ	-(SP)	:	
				7E	D4	001B9	CLRL	-(SP)	:	
				50	DD	001BB	PUSHL	R0	:	
				7E	D4	001BD	CLRL	-(SP)	:	
				57	DD	001BF	PUSHL	N	:	
		0000G		CF	9F	001C1	PUSHAB	PRINT_CSR_ERR	:	
		0000G		CF	9F	001C5	PUSHAB	UBIMOD_BDP	:	
	7E	0000G		CF	9A	001C9	MOVZBL	LUN, -(SP)	:	
				05	DD	001CE	PUSHL	#5	:	
				0A	FB	001D0	CALLS	#10, DS\$ERRHARD	:	
50	0000G	69	0003F464	8F	C9	001D3	BISL3	#259172, UNIBUS_SPACE, R0	:	0493
		6A		60	B0	001DD	MOVW	(R0), STATUS0	:	
		6A	FF1E	8F	AA	001E0	BICW2	#65310, STATUS0	:	
		50		6A	32	001E5	CVTWL	STATUS0, R0	:	
				1D	13	001E8	BEQL	8\$	:	
				7E	7C	001EA	CLRQ	-(SP)	:	
				50	DD	001EC	PUSHL	R0	:	
	7E			02	7D	001EE	MOVQ	#2, -(SP)	:	
		0000G		CF	9F	001F1	PUSHAB	FMT_UET_STAT	:	
		0000G		CF	9F	001F5	PUSHAB	PRINT_GENERAL	:	
		0000G		CF	9F	001F9	PUSHAB	UBIMOD_BDP	:	
	7E	0000G		CF	9A	001FD	MOVZBL	LUN, -(SP)	:	
				06	DD	00202	PUSHL	#6	:	
		69		0A	FB	00204	CALLS	#10, DS\$ERRHARD	:	
00001357		8F		6B	D1	00207	CMPL	SCRATCH, #4951	:	0494
				23	13	0020E	BEQL	9\$	:	
				7E	D4	00210	CLRL	-(SP)	:	0500
				6B	DD	00212	PUSHL	SCRATCH	:	
	7E	1357		8F	3C	00214	MOVZWL	#4951, -(SP)	:	
				57	DD	00219	PUSHL	N	:	
				03	DD	0021B	PUSHL	#3	:	
		0000G		CF	9F	0021D	PUSHAB	FMT_BDP_DATO	:	
		0000G		CF	9F	00221	PUSHAB	PRINT_GENERAL	:	
		0000G		CF	9F	00225	PUSHAB	UBIMOD_BDP	:	
	7E	0000G		CF	9A	00229	MOVZBL	LUN, -(SP)	:	
				07	DD	0022E	PUSHL	#7	:	
		69		0A	FB	00230	CALLS	#10, DS\$ERRHARD	:	
		50	04	AB	D0	00233	MOVL	SCRATCH+4, R0	:	0502
				20	13	00237	BEQL	10\$	:	
				7E	D4	00239	CLRL	-(SP)	:	0508
				50	DD	0023B	PUSHL	R0	:	
				7E	D4	0023D	CLRL	-(SP)	:	
				57	DD	0023F	PUSHL	N	:	
				03	DD	00241	PUSHL	#3	:	
		0000G		CF	9F	00243	PUSHAB	FMT_BDP_DATO	:	
		0000G		CF	9F	00247	PUSHAB	PRINT_GENERAL	:	
		0000G		CF	9F	0024B	PUSHAB	UBIMOD_BDP	:	
	7E	0000G		CF	9A	0024F	MOVZBL	LUN, -(SP)	:	


```

                                08 DD 00254      PUSHL #8
                                0A FB 00256      CALLS #10, DS$ERRHARD
                                ;
                                ; Change the data in the UET data register and change
                                ; Unibus address bit 1 in the UET address register from
                                ; zero to one.
                                ;
50      0000G CF 0003F462      8F C9 00259      10$: BISL3 #259170, UNIBUS_SPACE, R0      ; 0519
60      1234      8F B0 00263      MOVW #4660, (R0)      ; 0521
50      0000G CF 0003F460      8F C9 00268      BISL3 #259168, UNIBUS_SPACE, R0      ;
60      56      02 A1 00272      ADDW3 #2, UBUS_ADDR, (R0)      ; 0522
                                ;
                                ; Execute the third DATO and after a nominal wait check that th
                                ; e
                                ; second CMI destination now contains the expected data pattern
                                ;
50      0000G CF 0003F464      8F C9 00276      BISL3 #259172, UNIBUS_SPACE, R0      ; 0531
60      55      05 A9 00280      BISW3 #5, TEMP3, (R0)      ; 0533
7E      0A      7D 00284      MOVQ #10, -(SP)      ; 0535
00000000G 9F      02 FB 00287      CALLS #2, a#DS$WAITUS      ;
58      57      02 78 0028E      ASHL #2, N, R8      ; 0536
50      58      0000G CF C1 00292      ADDL3 UBI_UNDER_TEST, R8, R0      ;
04 AA      60 1FFFFFFE      8F CB 00298      BICL3 #536870910, (R0), STATUS1      ;
50      04      AA D0 002A1      MOVL STATUS1, R0      ;
1C      18 002A5      BGEQ 11$      ;
7E      7C 002A7      CLRQ -(SP)      ;
7E      D4 002A9      CLRL -(SP)      ;
50      DD 002AB      PUSHL R0      ;
7E      D4 002AD      CLRL -(SP)      ;
57      DD 002AF      PUSHL N      ;
0000G CF 9F 002B1      PUSHAB PRINT_CSR_ERR      ;
0000G CF 9F 002B5      PUSHAB UBIMOD_BDP      ;
7E      0000G CF 9A 002B9      MOVZBL LUN, -(SP)      ;
09      DD 002BE      PUSHL #9      ;
69      0A FB 002C0      11$: CALLS #10, DS$ERRHARD      ;
50      0000G CF 0003F464      8F C9 002C3      BISL3 #259172, UNIBUS_SPACE, R0      ; 0537
6A      60 B0 002CD      MOVW (R0), STATUS0      ;
6A      FF1E      8F AA 002D0      BICW2 #65310, STATUS0      ;
50      6A 32 002D5      CVTWL STATUS0, R0      ;
1D      13 002D8      BEQL 12$      ;
7E      7C 002DA      CLRQ -(SP)      ;
50      DD 002DC      PUSHL R0      ;
7E      02 7D 002DE      MOVQ #2, -(SP)      ;
0000G CF 9F 002E1      PUSHAB FMT_UET_STAT      ;
0000G CF 9F 002E5      PUSHAB PRINT_GENERAL      ;
0000G CF 9F 002E9      PUSHAB UBIMOD_BDP      ;
7E      0000G CF 9A 002ED      MOVZBL LUN, -(SP)      ;
0A      DD 002F2      PUSHL #10      ;
12345678 69      0A FB 002F4      12$: CALLS #10, DS$ERRHARD      ;
8F      04 AB D1 002F7      CMPL SCRATCH+4, #305419896      ; 0538
25      13 002FF      BEQL 13$      ;
7E      D4 00301      CLRL -(SP)      ; 0544
04 AB DD 00303      PUSHL SCRATCH+4      ;
12345678 8F DD 00306      PUSHL #305419896      ;

```

ZZ-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_016: Buffered Data Path DATO Test
TEST_016: Buffered Data Path DATO Test

G 10

6-Sep-1985

Fiche 2

Frame G10

Sequence 329

6-Sep-1985 17:20:57

VAX-11 Bliss-32 V4.1-746

Page 19

6-Sep-1985 17:15:01

[LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5

(3)

			57	DD	0030C	PUSHL	N	:
			03	DD	0030E	PUSHL	#3	:
		0000G	CF	9F	00310	PUSHAB	FMT_BDP_DATO	:
		0000G	CF	9F	00314	PUSHAB	PRINT_GENERAL	:
		0000G	CF	9F	00318	PUSHAB	UBIMOD_BDP	:
	7E	0000G	CF	9A	0031C	MOVZBL	LUN, -(SP)	:
			0B	DD	00321	PUSHL	#11	:
	69		0A	FB	00323	CALLS	#10, DS\$ERRHARD	:
	00000000G	9F	00	FB	00326	CALLS	#0, a#DS\$INLOOP	: 0550
		03	50	E9	0032D	BLBC	R0, 14\$	:
			FCEC	31	00330	BRW	2\$	:
FCE2	57		03	F1	00333	ACBL	#3, #1, N, 1\$	: 0354
	00000000G	9F	00	FB	00339	CALLS	#0, a#DS\$BREAK	: 0554
		50	01	D0	00340	MOVL	#1, R0	: 0556
				04	00343	RET		:

; Routine Size: 836 bytes, Routine Base: TEST_016 + 0000

```

: 0557 2 $DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Buffered Data Path DATOB Test');
: 0558 2
: 0559 2 !**
: 0560 2 ! TEST DESCRIPTION:
: 0561 2 !
: 0562 2 ! This test verifies the ability of the UBI to handle DATOB transactions
: 0563 2 ! through each of the buffered data paths (BDPs). The UBI behavior for
: 0564 2 ! a DATOB is primarily dependent on the contents of the buffer. The
: 0565 2 ! first case is when the buffer has Unibus data, no address match. The
: 0566 2 ! second case is when the buffer is empty or contains CMI data. The
: 0567 2 ! third case is when the buffer has Unibus data, and there is an address
: 0568 2 ! match.
: 0569 2 !
: 0570 2 ! The test algorithm is as follows. Purge the BDP, clear two CMI
: 0571 2 ! locations, load the UET data register with a unique pattern,
: 0572 2 ! then execute a DATOB. Change the data in the UET data register,
: 0573 2 ! execute another DATOB with a different destination address (it must be
: 0574 2 ! outside the original longword). After the first DATOB, check that the
: 0575 2 ! CMI destination is still at its initialized value. After the second
: 0576 2 ! DATOB, check that the CMI destination contains the first unique
: 0577 2 ! pattern. Execute a two more DATOBs at sequential byte addresses,
: 0578 2 ! changing the data in the UET data register for each DATOB, and
: 0579 2 ! check that the memory destination is still zero. Execute another
: 0580 2 ! DATOB with the next sequential byte address, and check that all four
: 0581 2 ! patterns, concatenated into a longword, were written into memory.
: 0582 2 ! This effectively tests the correct functionality of the byte flags.
: 0583 2 !
: 0584 2 ! ASSUMPTIONS:
: 0585 2 !
: 0586 2 ! Test 1-Test 15
: 0587 2 !--
: 0588 2
: 0589 2 REGISTER
: 0590 2     BUF_PFN,
: 0591 2     M,
: 0592 2     MAP_NUMBER,
: 0593 2     N,
: 0594 2     TEMP,
: 0595 2     TEMP3,
: 0596 2     UBUS_ADDR: WORD;
: 0597 2
: 0598 2 MACRO
: 0599 2     UBUS_PF = UBUS_ADDR<9,7>%;           ! Defines the Unibus page frame field
: 0600 2     UBUS_BO = UBUS_ADDR<0,9>%;           ! Defines the Unibus byte number field
: 0601 2
: 0602 2 CODECOMMENT
: 0603 2 ''
: 0604 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
: 0605 2 'a table of useable map entries. In the test which follows, the map',
: 0606 2 'selected is drawn from this table.',
: 0607 2 '';
: 0608 2 BEGIN
: 0609 2
: 0610 2     UNIBUS_SIZER();
: 0611 2

```



```

: 0612 2      END;
: 0613 2
: 0614 2      CODECOMMENT
: 0615 2      ''
: 0616 2      'Clear two successive longword aligned CMI locations. Put a unique data',
: 0617 2      'pattern into the UET data register. Set up the UBI for a DATOB',
: 0618 2      'addressing the first CMI longword, execute it, and check that both CMI',
: 0619 2      'destinations still contain zero. Change the UET address',
: 0620 2      'register to point to the second CMI longword, change the data pattern',
: 0621 2      'in the UET data pattern, and execute the second DATOB. After a',
: 0622 2      'nominal wait, check that the first unique data pattern is now in the',
: 0623 2      'first CMI destination, and then check that the second CMI destination',
: 0624 2      'is still empty.',
: 0625 2      '';
: 0626 3      BEGIN
: 0627 3
: 0628 3      INCR N FROM 1 TO 3 DO
: 0629 4          BEGIN
: 0630 4
: 0631 4              DO                      ! Scope loop begins here
: 0632 5                  BEGIN
: 0633 5
: 0634 5                  UBI_CSR(.N) = PURGE;
: 0635 5
: 0636 5                  CODECOMMENT
: 0637 5                  ''
: 0638 5                  'Set up the UET address register with the Unibus',
: 0639 5                  'address. Bits 9 to 15 (the Unibus page frame) contain',
: 0640 5                  'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
: 0641 5                  'the byte number field of the CMI address.',
: 0642 5                  '';
: 0643 6                      BEGIN
: 0644 6
: 0645 6                      SCRATCH[0] = 0;
: 0646 6                      SCRATCH[1] = 0;
: 0647 6                      UET_DATA_REG = %X'1357';
: 0648 6                      MAP_NUMBER = .FREE_MAP[.N];
: 0649 6                      UBUS_PF = .MAP_NUMBER;
: 0650 6                      UBUS_BO = SCRATCH[0];
: 0651 6                      UET_ADDR_REG = .UBUS_ADDR;
: 0652 6
: 0653 5                      END;
: 0654 5
: 0655 5                  CODECOMMENT
: 0656 5                  ''
: 0657 5                  'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
: 0658 5                  'map number.',
: 0659 5                  '';
: 0660 6                      BEGIN
: 0661 6
: 0662 6                      TEMP3 = .MAP_NUMBER<7,2> * 8;
: 0663 6
: 0664 5                      END;
: 0665 5
: 0666 5                  CODECOMMENT

```

```

; 0667 5
; 0668 5
; 0669 5
; 0670 5
; 0671 5
; 0672 5
; 0673 6
; 0674 6
; 0675 6
; 0676 6
; 0677 6
; 0678 6
; 0679 5
; 0680 5
; 0681 5
; 0682 5
; 0683 5
; 0684 5
; 0685 5
; 0686 6
; 0687 6
; 0688 6
; 0689 6
; 0690 6
; L 0691 6
; xPRINT:
; L 0692 7
; xPRINT:
; 0693 6
; 0694 6
; 0695 7
; P 0696 7
; P 0697 7
; P 0698 7
; L 0699 7
; xPRINT:
; 0700 6
; 0701 6
; 0702 6
; 0703 7
; P 0704 7
; P 0705 7
; P 0706 7
; L 0707 7
; xPRINT:
; 0708 6
; 0709 6
; 0710 5
; 0711 5
; 0712 5
; 0713 5
; 0714 5
; 0715 5
; 0716 5
; 0717 5

```

```

..
'Set up the UBI map. Put the CMI page frame number (PFN) into',
'the register BUF_PFN. Then call the map setup routine with the',
'parameters map number, PFN of the CMI buffer, no address',
'offset, and data path number.',
..
    BEGIN
        TEMP = SCRATCH[0];
        BUF_PFN = .TEMP<9,15>;
        MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.N);
    END;
CODECOMMENT
..
'Execute the first DATOB and after a nominal wait, check that',
'the two CMI destinations are still both cleared.',
..
    BEGIN
        UET_CTRL_REG = .TEMP3 OR DATOB;
        $DS_WAITUS(TIME=10);
        UBI_STATUS(.N,UBIMOD_BDP);          ! M7
        UET_STATUS(UBIMOD_BDP);             ! M7
        IF .SCRATCH[0] NEQ 0
        THEN
            BEGIN
                $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                    PRLINK = PRINT_GENERAL,
                    P1 = FMT_BDP_DATOB,P2 = 3,
                    P3 = .N,P4 = 0,P5 = .SCRATCH[0]);
                Test 17, Subtest 0, Error 3
            END;
            IF .SCRATCH[1] NEQ 0
            THEN
                BEGIN
                    $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                        PRLINK = PRINT_GENERAL,
                        P1 = FMT_BDP_DATOB,P2 = 3,
                        P3 = .N,P4 = 0,P5 = .SCRATCH[1]);
                    Test 17, Subtest 0, Error 4
                END;
            END;
        END;
CODECOMMENT
..
'Set up the UET address register with the Unibus',
'address. Bits 9 to 15 (the Unibus page frame) contain',
'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
'the byte number field of the CMI address.',

```

```

; 0718 5
; 0719 6
; 0720 6
; 0721 6
; 0722 6
; 0723 6
; 0724 6
; 0725 6
; 0726 5
; 0727 5
; 0728 5
; 0729 5
; 0730 5
; 0731 5
; 0732 5
; 0733 6
; 0734 6
; 0735 6
; 0736 6
; 0737 5
; 0738 5
; 0739 5
; 0740 5
; 0741 5
; 0742 5
; 0743 5
; 0744 5
; 0745 5
; 0746 6
; 0747 6
; 0748 6
; 0749 6
; 0750 6
; 0751 6
; 0752 5
; 0753 5
; 0754 5
; 0755 5
; 0756 5
; 0757 5
; 0758 5
; 0759 5
; 0760 6
; 0761 6
; 0762 6
; 0763 6
; 0764 6
; L 0765 6
; %PRINT: Test 17, Subtest 0, Error 5
; L 0766 7
; %PRINT: Test 17, Subtest 0, Error 6
; 0767 6
; 0768 6
; 0769 7
; P 0770 7

'' :
'' : BEGIN
'' :
'' : UET_DATA_REG = %X'FF78';
'' : UBUS_PF = .MAP_NUMBER;
'' : UBUS_BO = SCRATCH[1];
'' : UET_ADDR_REG = .UBUS_ADDR;
'' :
'' : END;
'' :
'' : CODECOMMENT
'' :
'' : 'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
'' : 'map number.',
'' :
'' : BEGIN
'' :
'' : TEMP3 = .MAP_NUMBER<7,2> * 8;
'' :
'' : END;
'' :
'' : CODECOMMENT
'' :
'' : 'Set up the UBI map. Put the CMI page frame number (PFN) into',
'' : 'the register BUF_PFN. Then call the map setup routine with the',
'' : 'parameters map number, PFN of the CMI buffer, no address',
'' : 'offset, and data path number.',
'' :
'' : BEGIN
'' :
'' : TEMP = SCRATCH[1];
'' : BUF_PFN = .TEMP<9,15>;
'' : MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.N);
'' :
'' : END;
'' :
'' : CODECOMMENT
'' :
'' : 'Execute another DATOB and after a nominal wait check that',
'' : 'the first CMI destination contains the expected data pattern.',
'' : 'Also, check that the second CMI destination is still cleared.',
'' :
'' : BEGIN
'' :
'' : UET_CTRL_REG = .TEMP3 OR DATOB;
'' :
'' : $DS_WAITUS(TIME=10);
'' : UBI_STATUS(.N,UBIMOD_BDP); ! M7
'' :
'' : UET_STATUS(UBIMOD_BDP); ! M7
'' :
'' : IF .SCRATCH[0] NEQ %X'57'
'' : THEN
'' : BEGIN
'' : $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,

```



```

; P 0771 7
; P 0772 7
; L 0773 7
; xPRINT:
; 0774 6
; 0775 6
; 0776 6
; 0777 7
; P 0778 7
; P 0779 7
; P 0780 7
; L 0781 7
; xPRINT:
; 0782 6
; 0783 6
; 0784 5
; 0785 5
; 0786 5
; 0787 5
; 0788 5
; 0789 5
; 0790 5
; 0791 6
; 0792 6
; 0793 6
; 0794 6
; 0795 6
; 0796 5
; 0797 5
; 0798 5
; 0799 5
; 0800 5
; 0801 5
; 0802 5
; 0803 6
; 0804 6
; 0805 6
; 0806 6
; 0807 6
; L 0808 6
; xPRINT:
; L 0809 7
; xPRINT:
; 0810 6
; 0811 6
; 0812 7
; P 0813 7
; P 0814 7
; P 0815 7
; L 0816 7
; xPRINT:
; 0817 6
; 0818 6
; 0819 5
; 0820 5

PRLINK = PRINT_GENERAL,
P1 = FMT_BDP_DATOB,P2 = 3,
P3 = .N,P4 = %X'57',P5 = .SCRATCH[0]);

Test 17, Subtest 0, Error 7
END;
IF .SCRATCH[1] NEQ 0
THEN
BEGIN
SDS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
PRLINK = PRINT_GENERAL,
P1 = FMT_BDP_DATOB,P2 = 3,
P3 = .N,P4 = 0,P5 = .SCRATCH[1]);

Test 17, Subtest 0, Error 8
END;

END;

CODECOMMENT
,,
'Change the data in the UET data register and change bit',
'0 of the Unibus address from a zero to a one.',
,,
BEGIN

UET_DATA_REG = %X'56FF';
UET_ADDR_REG = .UBUS_ADDR + 1;

END;

CODECOMMENT
,,
'Execute another DATOB and after a nominal wait check that the',
'second CMI destination is still empty.',
,,
BEGIN

UET_CTRL_REG = .TEMP3 OR DATOB;

SDS_WAITUS(TIME=10);
UBI_STATUS(.N,UBIMOD_BDP); ! M7

Test 17, Subtest 0, Error 9
UET_STATUS(UBIMOD_BDP); ! M7

Test 17, Subtest 0, Error 10
IF .SCRATCH[1] NEQ 0
THEN
BEGIN
SDS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
PRLINK = PRINT_GENERAL,
P1 = FMT_BDP_DATOB,P2 = 3,
P3 = .N,P4 = 0,P5 = .SCRATCH[1]);

Test 17, Subtest 0, Error 11
END;

END;

```

```

: 0821 5 CODECOMMENT
: 0822 5
: 0823 5 'Change the data in the UET data register and change the',
: 0824 5 'Unibus address to the next sequential byte address.',
: 0825 5
: 0826 6 BEGIN
: 0827 6
: 0828 6 UET_DATA_REG = %X'FF34';
: 0829 6 UET_ADDR_REG = .UBUS_ADDR + 2;
: 0830 6
: 0831 5 END;
: 0832 5
: 0833 5 CODECOMMENT
: 0834 5
: 0835 5 'Execute another DATOB and after a nominal wait check that the',
: 0836 5 'second CMI destination is still empty.',
: 0837 5
: 0838 6 BEGIN
: 0839 6
: 0840 6 UET_CTRL_REG = .TEMP3 OR DATOB;
: 0841 6
: 0842 6 $DS_WAITUS(TIME=10);
: L 0843 6 UBI_STATUS(.N,UBIMOD_BDP); ! M7
: %PRINT: Test 17, Subtest 0, Error 12
: L 0844 7 UET_STATUS(UBIMOD_BDP); ! M7
: %PRINT: Test 17, Subtest 0, Error 13
: 0845 6 IF .SCRATCH[1] NEQ 0
: 0846 6 THEN
: 0847 7 BEGIN
: P 0848 7 $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
: P 0849 7 PRLINK = PRINT_GENERAL,
: P 0850 7 P1 = FMT_BDP_DATOB,P2 = 3,
: L 0851 7 P3 = .N,P4 = 0,P5 = .SCRATCH[1]);
: %PRINT: Test 17, Subtest 0, Error 14
: 0852 6 END;
: 0853 6
: 0854 5 END;
: 0855 5
: 0856 5 CODECOMMENT
: 0857 5
: 0858 5 'Change the data in the UET data register and change the',
: 0859 5 'Unibus address to the next sequential byte address.',
: 0860 5
: 0861 6 BEGIN
: 0862 6
: 0863 6 UET_DATA_REG = %X'12FF';
: 0864 6 UET_ADDR_REG = .UBUS_ADDR + 3;
: 0865 6
: 0866 5 END;
: 0867 5
: 0868 5 CODECOMMENT
: 0869 5
: 0870 5 'Execute another DATOB and after a nominal wait check that the',
: 0871 5 'second CMI destination now contains 12345678.',
: 0872 5

```

```

; 0873 6          BEGIN
; 0874 6
; 0875 6          UET_CTRL_REG = .TEMP3 OR DATOB;
; 0876 6
; 0877 6          $DS_WAITUS(TIME=10);
; L 0878 6          UBI_STATUS(.N,UBIMOD_BDP);          ! M7
; xPRINT: Test 17, Subtest 0, Error 15
; L 0879 7          UET_STATUS(UBIMOD_BDP);          ! M7
; xPRINT: Test 17, Subtest 0, Error 16
; 0880 6          IF .SCRATCH[1] NEQ xX'12345678'
; 0881 6          THEN
; 0882 7          BEGIN
; P 0883 7          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
; P 0884 7          PRLINK = PRINT_GENERAL,
; P 0885 7          P1 = FMT_BDP_DATOB,P2 = 3,
; L 0886 7          P3 = .N,P4 = xX'12345678',P5 = .SCRATCH[1]);
; xPRINT: Test 17, Subtest 0, Error 17
; 0887 6          END;
; 0888 6
; 0889 5          END;
; 0890 5
; 0891 5          END
; 0892 4          WHILE $DS_INLOOP;          ! Scope loop ends here
; 0893 4
; 0894 3          END;
; 0895 3
; 0896 2          END;
; 0897 2
; 0898 1          $DS_ENDTEST;

```

```

                                .PSECT DISPATCH,NOWRT,NOEXE,2
                                .ADDRESS P.AAD, P.AAE, P.AAF, TEST_017
00000000' 00000000' 00000000' 00000000' 00018 $: .LONG 3, 0
                                .PSECT $PLIT$,NOWRT,NOEXE,2
                                .WORD 17
20 61 74 61 44 20 64 65 72 65 66 66 75 42 1D 00024 P.AAD: .ASCII <29>\Buffered Data Path DATOB Test\
74 73 65 54 20 42 4F 54 41 44 20 68 74 61 50 00026 P.AAE:
                                .LONG 0
                                .PSECT TEST_017,NOWRT,2
                                .ENTRY TEST_017, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0557
                                OFFC 00000
                                MOVAB UNIBUS_SPACE, R11
                                MOVAB STATUS0, R10
                                MOVAB a#DS$ERRHARD, R9
                                ; Call the Unibus sizer routine to locate any Unibus memory. T

```


; ;
;
;
;
;
;
;
;
;

0000G CF 00 FB 00013

```

his builds
; a table of useable map entries. In the test which follows, t
he map
; selected is drawn from this table.
;
CALLS #0, UNIBUS_SIZER ; 0610
;
; Clear two successive longword aligned CMI locations. Put a u
nique data
; pattern into the UET data register. Set up the UBI for a DAT
OB
; addressing the first CMI longword, execute it, and check that
both CMI
; destinations still contain zero. Change the UET address
; register to point to the second CMI longword, change the data
pattern
; in the UET data pattern, and execute the second DATOB. After
a
; nominal wait, check that the first unique data pattern is now
in the
; first CMI destination, and then check that the second CMI des
tination
; is still empty.

```

	57		01	D0	00018
58	57		02	78	0001B
50	58	0000G	CF	C1	0001F
	60		01	D0	00025

```

1$:      MOVL      #1, N                                ; 0628
2$:      ASHL      #2, N, R8                            ; 0634
3$:      ADDL3     UBI_UNDER_TEST, R8, R0
4$:      MOVL      #1, (R0)

```

```
; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.
```

			0000G	CF	7C	00028
	50	6B	0003F462	8F	C9	0002C
		60	1357	8F	B0	00034
		53	0000GCF	47	3C	00039
56	07	09		53	F0	0003F
		50	0000G	CF	9E	00044
56	09	00		50	F0	00049
	50	6B	0003F460	8F	C9	0004E
		60		56	B0	00056

```

CLRQ    SCRATCH                                : 0645
BISL3   #259170, UNIBUS_SPACE, R0             : 0646
MOVW    #4951, (R0)                            : 0647
MOVZWL  FREE_MAP[N], MAP_NUMBER               : 0648
INSV    MAP_NUMBER, #9, #7, UBUS_ADDR         : 0649
MOVAB   SCRATCH, R0                           : 0650
INSV    R0, #0, #9, UBUS_ADDR                 :
BISL3   #259168, UNIBUS_SPACE, R0             :
MOVW    UBUS_ADDR, (R0)                       : 0651

```

```
; Write the 2 MSBs of the Unibus address with the 2 MSBs of the
; map number.
```

55	53	02	07	EF	00059
		55	08	C4	0005E

```
EXTZV  #7, #2, MAP_NUMBER, TEMP3      : 0662
MULL2  #8, TEMP3                        :
```

```
; Set up the UBI map. Put the CMI page frame number (PFN) into
; the register BUF_PFN. Then call the map setup routine with t
; he
; parameters map number, PFN of the CMI buffer, no address
; offset, and data path number.
```

```

52          54          54          0000G  CF  9E 00061  ;      MOVAB  SCRATCH, TEMP          ; 0675
          54          0F          09  EF 00066  ;      EXTZV  #9, #15, TEMP, BUF_PFN  ; 0676
          57          DD 0006B  ;      PUSHL  N                      ; 0677
          7E          D4 0006D  ;      CLRL   -(SP)                  ;
          52          DD 0006F  ;      PUSHL  BUF_PFN                  ;
          53          DD 00071  ;      PUSHL  MAP_NUMBER                 ;
          04          FB 00073  ;      CALLS  #4, MAP_SETUP              ;
          0000G  CF          04          ;
          ; Execute the first DATOB and after a nominal wait, check that
          ; the two CMI destinations are still both cleared.
          ;
          50          6B 0003F464 8F  C9 00078  ;      BISL3  #259172, UNIBUS_SPACE, R0  ; 0686
          60          55          07  A9 00080  ;      BISW3  #7, TEMP3, (R0)          ; 0688
          7E          0A 7D 00084  ;      MOVQ   #10, -(SP)              ; 0690
          00000000G 9F          02  FB 00087  ;      CALLS  #2, a#DS$WAITUS        ;
          04  AA          0000GDF47 1FFFFFFE 8F  CB 0008E  ;      BICL3  #536870910, aUBI_UNDER_TEST[N], STATUS1  ; 0691
          50          04          AA  D0 0009A  ;      MOVL   STATUS1, R0              ;
          1C          18 0009E  ;      BGEQ     3$                      ;
          7E          7C 000A0  ;      CLRQ   -(SP)                  ;
          7E          D4 000A2  ;      CLRL   -(SP)                  ;
          50          DD 000A4  ;      PUSHL  R0                      ;
          7E          D4 000A6  ;      CLRL   -(SP)                  ;
          57          DD 000A8  ;      PUSHL  N                      ;
          0000G  CF  9F 000AA  ;      PUSHAB PRINT_CSR_ERR          ;
          0000G  CF  9F 000AE  ;      PUSHAB UBIMOD_BDP            ;
          7E          0000G  CF  9A 000B2  ;      MOVZBL LUN, -(SP)              ;
          01          DD 000B7  ;      PUSHL  #1                      ;
          69          0A  FB 000B9  ;      CALLS  #10, DS$ERRHARD          ;
          50          6B 0003F464 8F  C9 000BC 3$:  ;      BISL3  #259172, UNIBUS_SPACE, R0  ; 0692
          6A          60  B0 000C4  ;      MOVW    (R0), STATUS0          ;
          6A          FF1E 8F  AA 000C7  ;      BICW2  #65310, STATUS0        ;
          50          6A  32 000CC  ;      CVTWL  STATUS0, R0              ;
          1D          13 000CF  ;      BEQL   4$                      ;
          7E          7C 000D1  ;      CLRQ   -(SP)                  ;
          50          DD 000D3  ;      PUSHL  R0                      ;
          7E          02  7D 000D5  ;      MOVQ   #2, -(SP)              ;
          0000G  CF  9F 000D8  ;      PUSHAB FMT_UET_STAT          ;
          0000G  CF  9F 000DC  ;      PUSHAB PRINT_GENERAL          ;
          0000G  CF  9F 000E0  ;      PUSHAB UBIMOD_BDP            ;
          7E          0000G  CF  9A 000E4  ;      MOVZBL LUN, -(SP)              ;
          02          DD 000E9  ;      PUSHL  #2                      ;
          69          0A  FB 000EB  ;      CALLS  #10, DS$ERRHARD          ;
          50          0000G  CF  D0 000EE 4$:  ;      MOVL   SCRATCH, R0              ; 0693
          20          13 000F3  ;      BEQL   5$                      ;
          7E          D4 000F5  ;      CLRL   -(SP)                  ;
          50          DD 000F7  ;      PUSHL  R0                      ;
          7E          D4 000F9  ;      CLRL   -(SP)                  ;
          57          DD 000FB  ;      PUSHL  N                      ;
          03          DD 000FD  ;      PUSHL  #3                      ;
          0000G  CF  9F 000FF  ;      PUSHAB FMT_BDP_DATOB          ;
          0000G  CF  9F 00103  ;      PUSHAB PRINT_GENERAL          ;
          0000G  CF  9F 00107  ;      PUSHAB UBIMOD_BDP            ;
          7E          0000G  CF  9A 0010B  ;      MOVZBL LUN, -(SP)              ;
          03          DD 00110  ;      PUSHL  #3                      ;

```

```

        69      0A FB 00112      CALLS #10, DS$ERRHARD      ;
        50      0000G CF D0 00115 5$:      MOVL SCRATCH+4, R0      ; 0701
        20      13 0011A      BEQL 6$      ;
        7E      D4 0011C      CLRL -(SP)      ; 0707
        50      DD 0011E      PUSHL R0      ;
        7E      D4 00120      CLRL -(SP)      ;
        57      DD 00122      PUSHL N      ;
        03      DD 00124      PUSHL #3      ;
        0000G CF 9F 00126      PUSHAB FMT_BDP_DATOB      ;
        0000G CF 9F 0012A      PUSHAB PRINT_GENERAL      ;
        0000G CF 9F 0012E      PUSHAB UBIMOD_BDP      ;
        7E      0000G CF 9A 00132      MOVZBL LUN, -(SP)      ;
        04      DD 00137      PUSHL #4      ;
        69      0A FB 00139      CALLS #10, DS$ERRHARD      ;
;
; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.
;
        50      6B 0003F462 8F C9 0013C 6$:      BISL3 #259170, UNIBUS_SPACE, R0      ; 0719
        60      FF78      8F B0 00144      MOVW #-136, (R0)      ; 0721
        56      07      09      53 F0 00149      INSV MAP_NUMBER, #9, #7, UNIBUS_ADDR      ; 0722
        50      0000G CF 9E 0014E      MOVAB SCRATCH+4, R0      ; 0723
        56      09      00      50 F0 00153      INSV R0, #0, #9, UNIBUS_ADDR      ;
        50      6B 0003F460 8F C9 00158      BISL3 #259168, UNIBUS_SPACE, R0      ;
        60      56 B0 00160      MOVW UNIBUS_ADDR, (R0)      ; 0724
;
; Write the 2 MSBs of the Unibus address with the 2 MSBs of the
; map number.
;
        55      53      02      07 EF 00163      EXTZV #7, #2, MAP_NUMBER, TEMP3      ; 0735
        55      08 C4 00168      MULL2 #8, TEMP3      ;
;
; Set up the UBI map. Put the CMI page frame number (PFN) into
; the register BUF_PFN. Then call the map setup routine with t
; he
; parameters map number, PFN of the CMI buffer, no address
; offset, and data path number.
;
        52      54      54      0000G CF 9E 0016B      MOVAB SCRATCH+4, TEMP      ; 0748
        0F      09 EF 00170      EXTZV #9, #15, TEMP, BUF_PFN      ; 0749
        57      DD 00175      PUSHL N      ; 0750
        7E      D4 00177      CLRL -(SP)      ;
        52      DD 00179      PUSHL BUF_PFN      ;
        53      DD 0017B      PUSHL MAP_NUMBER      ;
        0000G CF 04 FB 0017D      CALLS #4, MAP_SETUP      ;
;
; Execute another DATOB and after a nominal wait check that
; the first CMI desination contains the expected data pattern.
; Also, check that the second CMI destination is still cleared.
;
        50      6B 0003F464 8F C9 00182      BISL3 #259172, UNIBUS_SPACE, R0      ; 0760
        60      55      07 A9 0018A      BISW3 #7, TEMP3, (R0)      ; 0762
        7E      0A 7D 0018E      MOVQ #10, -(SP)      ; 0764

```


ZZ-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_017: Buffered Data Path DATOB Test
TEST_017: Buffered Data Path DATOB Test

E 11

6-Sep-1985

Fiche 2 Frame E11

Sequence 340

6-Sep-1985 17:20:57

VAX-11 Bliss-32 V4.1-746

Page 30

6-Sep-1985 17:15:01

[LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5

(4)

04	AA	00000000G	9F	02	FB	00191	CALLS	#2, a#DS\$WAITUS	:	
		0000GDF47	1FFFFFFE	8F	CB	00198	BICL3	#536870910, aUBI_UNDER_TEST[N], STATUS1	:	0765
		50	04	AA	D0	001A4	MOVL	STATUS1, R0	:	
				1C	18	001A8	BGEQ	7\$	:	
				7E	7C	001AA	CLRQ	-(SP)	:	
				7E	D4	001AC	CLRL	-(SP)	:	
				50	DD	001AE	PUSHL	R0	:	
				7E	D4	001B0	CLRL	-(SP)	:	
				57	DD	001B2	PUSHL	N	:	
		0000G		CF	9F	001B4	PUSHAB	PRINT_CSR_ERR	:	
		0000G		CF	9F	001B8	PUSHAB	UBIMOD_BDP	:	
	7E	0000G		CF	9A	001BC	MOVZBL	LUN, -(SP)	:	
				05	DD	001C1	PUSHL	#5	:	
	69			0A	FB	001C3	CALLS	#10, DS\$ERRHARD	:	
50	6B	0003F464		8F	C9	001C6	BISL3	#259172, UNIBUS_SPACE, R0	:	0766
	6A			60	B0	001CE	MOVW	(R0), STATUS0	:	
	6A	FF1E		8F	AA	001D1	BICW2	#65310, STATUS0	:	
	50			6A	32	001D6	CVTL	STATUS0, R0	:	
				1D	13	001D9	BEQL	8\$	:	
				7E	7C	001DB	CLRQ	-(SP)	:	
				50	DD	001DD	PUSHL	R0	:	
	7E			02	7D	001DF	MOVQ	#2, -(SP)	:	
		0000G		CF	9F	001E2	PUSHAB	FMT_UET_STAT	:	
		0000G		CF	9F	001E6	PUSHAB	PRINT_GENERAL	:	
		0000G		CF	9F	001EA	PUSHAB	UBIMOD_BDP	:	
	7E	0000G		CF	9A	001EE	MOVZBL	LUN, -(SP)	:	
				06	DD	001F3	PUSHL	#6	:	
	69			0A	FB	001F5	CALLS	#10, DS\$ERRHARD	:	
	8F	0000G		CF	D1	001F8	CMPL	SCRATCH, #87	:	0767
				24	13	00201	BEQL	9\$	:	
				7E	D4	00203	CLRL	-(SP)	:	0773
		0000G		CF	DD	00205	PUSHL	SCRATCH	:	
	7E	57		8F	9A	00209	MOVZBL	#87, -(SP)	:	
				57	DD	0020D	PUSHL	N	:	
				03	DD	0020F	PUSHL	#3	:	
		0000G		CF	9F	00211	PUSHAB	FMT_BDP_DATOB	:	
		0000G		CF	9F	00215	PUSHAB	PRINT_GENERAL	:	
		0000G		CF	9F	00219	PUSHAB	UBIMOD_BDP	:	
	7E	0000G		CF	9A	0021D	MOVZBL	LUN, -(SP)	:	
				07	DD	00222	PUSHL	#7	:	
	69			0A	FB	00224	CALLS	#10, DS\$ERRHARD	:	
	50	0000G		CF	D0	00227	MOVL	SCRATCH+4, R0	:	0775
				20	13	0022C	BEQL	10\$	:	
				7E	D4	0022E	CLRL	-(SP)	:	0781
				50	DD	00230	PUSHL	R0	:	
				7E	D4	00232	CLRL	-(SP)	:	
				57	DD	00234	PUSHL	N	:	
				03	DD	00236	PUSHL	#3	:	
		0000G		CF	9F	00238	PUSHAB	FMT_BDP_DATOB	:	
		0000G		CF	9F	0023C	PUSHAB	PRINT_GENERAL	:	
		0000G		CF	9F	00240	PUSHAB	UBIMOD_BDP	:	
	7E	0000G		CF	9A	00244	MOVZBL	LUN, -(SP)	:	
				08	DD	00249	PUSHL	#8	:	
	69			0A	FB	0024B	CALLS	#10, DS\$ERRHARD	:	

; Change the data in the UET data register and change bit
; 0 of the Unibus address from a zero to a one.

50	6B	0003F462	8F	C9	0024E	10\$:	BISL3	#259170, UNIBUS_SPACE, R0	; 0791
	60	56FF	8F	B0	00256		MOVW	#22271, (R0)	; 0793
50	6B	0003F460	8F	C9	0025B		BISL3	#259168, UNIBUS_SPACE, R0	
60	56		01	A1	00263		ADDW3	#1, UBUS_ADDR, (R0)	; 0794
; Execute another DATOB and after a nominal wait check that the									
; second CMI destination is still empty.									
50	6B	0003F464	8F	C9	00267		BISL3	#259172, UNIBUS_SPACE, R0	; 0803
60	55		07	A9	0026F		BISW3	#7, TEMP3, (R0)	; 0805
	7E		0A	7D	00273		MOVQ	#10, -(SP)	; 0807
	9F		02	FB	00276		CALLS	#2, aDS\$WAITUS	
04	AA	00000000G	8F	CB	0027D		BICL3	#536870910, aUBI_UNDER_TEST[N], STATUS1	; 0808
		0000GDF47 1FFFFFFE	AA	D0	00289		MOVL	STATUS1, R0	
		04	1C	18	0028D		BGEQ	11\$	
			7E	7C	0028F		CLRQ	-(SP)	
			7E	D4	00291		CLRL	-(SP)	
			50	DD	00293		PUSHL	R0	
			7E	D4	00295		CLRL	-(SP)	
			57	DD	00297		PUSHL	N	
		0000G	CF	9F	00299		PUSHAB	PRINT_CSR_ERR	
		0000G	CF	9F	0029D		PUSHAB	UBIMOD_BDP	
	7E	0000G	CF	9A	002A1		MOVZBL	LUN, -(SP)	
			09	DD	002A6		PUSHL	#9	
	69		0A	FB	002A8		CALLS	#10, DS\$ERRHARD	
50	6B	0003F464	8F	C9	002AB	11\$:	BISL3	#259172, UNIBUS_SPACE, R0	; 0809
	6A		60	B0	002B3		MOVW	(R0), STATUS0	
	6A	FF1E	8F	AA	002B6		BICW2	#65310, STATUS0	
	50		6A	32	002BB		CVTWL	STATUS0, R0	
			1D	13	002BE		BEQL	12\$	
			7E	7C	002C0		CLRQ	-(SP)	
			50	DD	002C2		PUSHL	R0	
	7E		02	7D	002C4		MOVQ	#2, -(SP)	
		0000G	CF	9F	002C7		PUSHAB	FMT_UET_STAT	
		0000G	CF	9F	002CB		PUSHAB	PRINT_GENERAL	
		0000G	CF	9F	002CF		PUSHAB	UBIMOD_BDP	
	7E	0000G	CF	9A	002D3		MOVZBL	LUN, -(SP)	
			0A	DD	002D8		PUSHL	#10	
	69		0A	FB	002DA		CALLS	#10, DS\$ERRHARD	
50		0000G	CF	D0	002DD	12\$:	MOVL	SCRATCH+4, R0	; 0810
			20	13	002E2		BEQL	13\$	
			7E	D4	002E4		CLRL	-(SP)	; 0816
			50	DD	002E6		PUSHL	R0	
			7E	D4	002E8		CLRL	-(SP)	
			57	DD	002EA		PUSHL	N	
			03	DD	002EC		PUSHL	#3	
		0000G	CF	9F	002EE		PUSHAB	FMT_BDP_DATOB	
		0000G	CF	9F	002F2		PUSHAB	PRINT_GENERAL	
		0000G	CF	9F	002F6		PUSHAB	UBIMOD_BDP	
	7E	0000G	CF	9A	002FA		MOVZBL	LUN, -(SP)	
			0B	DD	002FF		PUSHL	#11	
69			0A	FB	00301		CALLS	#10, DS\$ERRHARD	

```

; Change the data in the UET data register and change the
; Unibus address to the next sequential byte address.
;
50      6B 0003F462 8F C9 00304 13$: BISL3 #259170, UNIBUS_SPACE, R0 ; 0826
60      60 FF34 8F B0 0030C MOVW #-204, (R0) ; 0828
50      6B 0003F460 8F C9 00311 BISL3 #259168, UNIBUS_SPACE, R0 ;
60      56 02 A1 00319 ADDW3 #2, UNIBUS_ADDR, (R0) ; 0829
;
; Execute another DATOB and after a nominal wait check that the
; second CMI destination is still empty.
;
50      6B 0003F464 8F C9 0031D BISL3 #259172, UNIBUS_SPACE, R0 ; 0838
60      55 07 A9 00325 BISM3 #7, TEMP3, (R0) ; 0840
7E      0A 7D 00329 MOVQ #10, -(SP) ; 0842
04 AA 00000000G 9F 02 FB 0032C CALLS #2, a#DS$WAITUS ;
0000GDF47 1FFFFFFE 8F CB 00333 BICL3 #536870910, aUBI_UNDER_TEST[N], STATUS1 ; 0843
50      04 AA 50 04 AA D0 0033F MOVL STATUS1, R0 ;
1C 18 00343 BGEQ 14$ ;
7E 7C 00345 CLRQ -(SP) ;
7E D4 00347 CLRL -(SP) ;
50 DD 00349 PUSHL R0 ;
7E D4 0034B CLRL -(SP) ;
57 DD 0034D PUSHL N ;
0000G CF 9F 0034F PUSHAB PRINT_CSR_ERR ;
0000G CF 9F 00353 PUSHAB UBIMOD_BDP ;
7E 0000G CF 9A 00357 MOVZBL LUN, -(SP) ;
0C DD 0035C PUSHL #12 ;
69 0A FB 0035E CALLS #10, DS$ERRHARD ;
50 6B 0003F464 8F C9 00361 14$: BISL3 #259172, UNIBUS_SPACE, R0 ; 0844
6A 60 B0 00369 MOVW (R0), STATUS0 ;
6A FF1E 8F AA 0036C BICW2 #65310, STATUS0 ;
50 6A 32 00371 CVTWL STATUS0, R0 ;
1D 13 00374 BEQL 15$ ;
7E 7C 00376 CLRQ -(SP) ;
50 DD 00378 PUSHL R0 ;
7E 02 7D 0037A MOVQ #2, -(SP) ;
0000G CF 9F 0037D PUSHAB FMT_UET_STAT ;
0000G CF 9F 00381 PUSHAB PRINT_GENERAL ;
0000G CF 9F 00385 PUSHAB UBIMOD_BDP ;
7E 0000G CF 9A 00389 MOVZBL LUN, -(SP) ;
0D DD 0038E PUSHL #13 ;
69 0A FB 00390 CALLS #10, DS$ERRHARD ;
50 0000G CF D0 00393 15$: MOVL SCRATCH+4, R0 ; 0845
20 13 00398 BEQL 16$ ;
7E D4 0039A CLRL -(SP) ;
50 DD 0039C PUSHL R0 ;
7E D4 0039E CLRL -(SP) ;
57 DD 003A0 PUSHL N ;
03 DD 003A2 PUSHL #3 ;
0000G CF 9F 003A4 PUSHAB FMT_BDP_DATOB ;
0000G CF 9F 003A8 PUSHAB PRINT_GENERAL ;
0000G CF 9F 003AC PUSHAB UBIMOD_BDP ;
7E 0000G CF 9A 003B0 MOVZBL LUN, -(SP) ;
0E DD 003B5 PUSHL #14 ;

```



```

69          0A FB 003B7          CALLS #10, DS$ERRHARD          ;
; Change the data in the UET data register and change the
; Unibus address to the next sequential byte address.
;
50          6B 0003F462 8F C9 003BA 16$: BISL3 #259170, UNIBUS_SPACE, R0 ; 0861
60          60 12FF 8F B0 003C2 MOVW #4863, (R0) ; 0863
50          6B 0003F460 8F C9 003C7 BISL3 #259168, UNIBUS_SPACE, R0 ;
60          56 03 A1 003CF ADDW3 #3, UBUS_ADDR, (R0) ; 0864
; Execute another DATOB and after a nominal wait check that the
; second CMI destination now contains 12345678.
;
50          6B 0003F464 8F C9 003D3 BISL3 #259172, UNIBUS_SPACE, R0 ; 0873
60          55 07 A9 003DB BISW3 #7, TEMP3, (R0) ; 0875
7E          0A 7D 003DF MOVQ #10, -(SP) ; 0877
00000000G 9F 02 FB 003E2 CALLS #2, a#DS$WAITUS ;
58          57 02 78 003E9 ASHL #2, N, R8 ; 0878
50          58 0000G CF C1 003ED ADDL3 UBI_UNDER_TEST, R8, R0 ;
04 AA 60 1FFFFFFE 8F CB 003F3 BICL3 #536870910, (R0), STATUS1 ;
50          50 04 AA D0 003FC MOVL STATUS1, R0 ;
1C 18 00400 BGEQ 17$ ;
7E 7C 00402 CLRQ -(SP) ;
7E D4 00404 CLRL -(SP) ;
50 DD 00406 PUSHL R0 ;
7E D4 00408 CLRL -(SP) ;
57 DD 0040A PUSHL N ;
0000G CF 9F 0040C PUSHAB PRINT_CSR_ERR ;
0000G CF 9F 00410 PUSHAB UBIMOD_BDP ;
7E 0000G CF 9A 00414 MOVZBL LUN, -(SP) ;
0F DD 00419 PUSHL #15 ;
69 0A FB 0041B CALLS #10, DS$ERRHARD ;
50 6B 0003F464 8F C9 0041E 17$: BISL3 #259172, UNIBUS_SPACE, R0 ; 0879
6A 60 B0 00426 MOVW (R0), STATUS0 ;
6A FF1E 8F AA 00429 BICW2 #65310, STATUS0 ;
50 6A 32 0042E CVTWL STATUS0, R0 ;
1D 13 00431 BEQL 18$ ;
7E 7C 00433 CLRQ -(SP) ;
50 DD 00435 PUSHL R0 ;
7E 02 7D 00437 MOVQ #2, -(SP) ;
0000G CF 9F 0043A PUSHAB FMT_UET_STAT ;
0000G CF 9F 0043E PUSHAB PRINT_GENERAL ;
0000G CF 9F 00442 PUSHAB UBIMOD_BDP ;
7E 0000G CF 9A 00446 MOVZBL LUN, -(SP) ;
10 DD 0044B PUSHL #16 ;
69 0A FB 0044D CALLS #10, DS$ERRHARD ;
12345678 8F 0000G CF D1 00450 18$: CMPL SCRATCH+4, #305419896 ; 0880
26 13 00459 BEQL 19$ ;
7E D4 0045B CLRL -(SP) ; 0886
0000G CF DD 0045D PUSHL SCRATCH+4 ;
12345678 8F DD 00461 PUSHL #305419896 ;
57 DD 00467 PUSHL N ;
03 DD 00469 PUSHL #3 ;
0000G CF 9F 0046B PUSHAB FMT_BDP_DATOB ;
0000G CF 9F 0046F PUSHAB PRINT_GENERAL ;

```

ZZ-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_017: Buffered Data Path DATOB Test
TEST_017: Buffered Data Path DATOB Test

I 11
6-Sep-1985
6-Sep-1985 17:20:57
6-Sep-1985 17:15:01

Fiche 2
Frame 111
VAX-11 Bliss-32 V4.1-746
[LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5

Sequence 344
Page 34
(4)

		0000G	CF	9F	00473	PUSHAB	UBIMOD_BDP	:
	7E	0000G	CF	9A	00477	MOVZBL	LUN, -(SP)	:
			11	DD	0047C	PUSHL	#17	:
	69		0A	FB	0047E	CALLS	#10, DS\$ERRHARD	:
	00000000G	9F	00	FB	00481	CALLS	#0, a#DS\$INLOOP	: 0892
	03		50	E9	00488	BLBC	R0, 20\$	:
			FB91	31	0048B	BRW	2\$	:
FB87	57		03	F1	0048E	ACBL	#3, #1, N, 1\$	: 0628
	00000000G	9F	00	FB	00494	CALLS	#0, a#DS\$BREAK	: 0896
		50	01	D0	0049B	MOVL	#1, R0	: 0898
			04	0049E	RET			:

; Routine Size: 1183 bytes, Routine Base: TEST_017 + 0000

```

; 0899 2 $DS_BGNTTEST (SECTION=(DEFAULT,ALL),TEXT='Buffered Data Path Autopurge Test');
; 0900 2
; 0901 2 !++
; 0902 2 ! TEST DESCRIPTION:
; 0903 2 !
; 0904 2 ! This tests the ability of the UBI to perform autopurges with each
; 0905 2 ! buffered data path.
; 0906 2 !
; 0907 2 ! An autopurge should occur in two main cases:
; 0908 2 !
; 0909 2 ! 1. For a DATI following a DATO (which caused no CMI write).
; 0910 2 ! 2. For a DATO following a DATO (which caused no CMI write),
; 0911 2 ! with the address of the second DATO outside the longword
; 0912 2 ! containing the first DATO.
; 0913 2 !
; 0914 2 ! The second case of autopurge is tested in the BDP DATO test. This
; 0915 2 ! test will deal with the first case.
; 0916 2 !
; 0917 2 ! The test algorithm is as follows. Purge the BDP and clear a longword
; 0918 2 ! in memory. Put a unique pattern in the UET data register. Execute a
; 0919 2 ! DATO, then check that the memory location is still empty. Next, set
; 0920 2 ! up for a DATI with another unique pattern. Execute the DATI, and
; 0921 2 ! check that the memory location now contains the pattern from the DATO.
; 0922 2 ! Check also that the DATI worked.
; 0923 2 !
; 0924 2 ! TEST ASSUMPTIONS:
; 0925 2 !
; 0926 2 ! Test 1-Test 16
; 0927 2 !--
; 0928 2
; 0929 2 REGISTER
; 0930 2 BUF_PFN,
; 0931 2 M,
; 0932 2 MAP_NUMBER,
; 0933 2 TEMP,
; 0934 2 TEMP1: WORD,
; 0935 2 TEMP3,
; 0936 2 UBUS_ADDR: WORD;
; 0937 2
; 0938 2 MACRO
; 0939 2 UBUS_PF = UBUS_ADDR<9,7> x,
; 0940 2 UBUS_BO = UBUS_ADDR<0,9> x;
; 0941 2
; 0942 2 CODECOMMENT
; 0943 2 ''
; 0944 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds a',
; 0945 2 'table of useable map entries. In the test which follows, the map selected',
; 0946 2 'is drawn from this table.',
; 0947 2 ''
; 0948 3 BEGIN
; 0949 3 UNIBUS_SIZER();
; 0950 2 END;
; 0951 2
; 0952 2 INCR N FROM 1 TO 3 DO
; 0953 3 BEGIN

```



```

; 0954 3
; 0955 3      DO                      ! Scope loop begins here
; 0956 4      BEGIN
; 0957 4
; 0958 4      CODECOMMENT
; 0959 4      ;;
; 0960 4      'Purge the BDP. Clear a longword aligned location in memory. Put a',
; 0961 4      'unique pattern into the UET data register.',
; 0962 4      ;;
; 0963 5          BEGIN
; 0964 5
; 0965 5          UBI_CSR(.N) = PURGE;
; 0966 5          $DS_WAITUS(TIME=10);          ! Give the purge time to work
; 0967 5          SCRATCH[0] = 0;
; 0968 5
; 0969 4          END;
; 0970 4
; 0971 4      CODECOMMENT
; 0972 4      ;;
; 0973 4      'Set up the UET address register with the Unibus address. Bits',
; 0974 4      '9 to 15 (the Unibus page frame) contain the 7 LSBs of the map',
; 0975 4      'number. Bits 0 to 8 contain the byte number field of the CMI',
; 0976 4      'address.',
; 0977 4      ;;
; 0978 5          BEGIN
; 0979 5
; 0980 5          UET_DATA_REG = %X'1234';
; 0981 5          MAP_NUMBER = .FREE_MAP(.N);
; 0982 5          UBUS_PF = .MAP_NUMBER;
; 0983 5          UBUS_BO = SCRATCH[0];
; 0984 5          UET_ADDR_REG = .UBUS_ADDR;
; 0985 5
; 0986 4          END;
; 0987 4
; 0988 4      CODECOMMENT
; 0989 4      ;;
; 0990 4      'Write the 2 MSBs of the Unibus address with the 2 MSBs of the',
; 0991 4      'map number.',
; 0992 4      ;;
; 0993 5          BEGIN
; 0994 5
; 0995 5              TEMP3 = .MAP_NUMBER<7,2> * 8;
; 0996 5
; 0997 4          END;
; 0998 4
; 0999 4      CODECOMMENT
; 1000 4      ;;
; 1001 4      'Set up the UBI map. Put the CMI page frame number (PFN) into the',
; 1002 4      'register BUF_PFN. Then call the map setup routine with the',
; 1003 4      'parameters map number, PFN of the CMI buffer, no byte offset, and',
; 1004 4      'data path number.',
; 1005 4      ;;
; 1006 5          BEGIN
; 1007 5
; 1008 5              TEMP = SCRATCH[0];

```

```

; 1009 5      BUF_PFN = .TEMP<9,15>;
; 1010 5      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.N);
; 1011 5
; 1012 4      END;
; 1013 4
; 1014 4      CODECOMMENT
; 1015 4      '
; 1016 4      'Execute a DATO, and after a nominal wait, check that the memory',
; 1017 4      'location still contains zero.',
; 1018 4      '
; 1019 5      BEGIN
; 1020 5
; 1021 5      UET_CTRL_REG = .TEMP3 OR DATO;
; 1022 5
; 1023 5      $DS_WAITUS(TIME=10);
; L 1024 5      UBI_STATUS(.N,UBIMOD_BDP);          ! M7
; %PRINT:      Test 18, Subtest 0, Error 1
; L 1025 6      UET_STATUS(UBIMOD_BDP);          ! M7
; %PRINT:      Test 18, Subtest 0, Error 2
; 1026 5      IF .SCRATCH[0] NEQ 0
; 1027 5      THEN
; 1028 6      BEGIN
; P 1029 6      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
; P 1030 6      PRLINK = PRINT_GENERAL,
; P 1031 6      P1 = FMT_BDP_DATO,P2 = 3,
; L 1032 6      P3 = .N,P4 = 0,P5 = .SCRATCH[0]);
; %PRINT:      Test 18, Subtest 0, Error 3
; 1033 5      END;
; 1034 5
; 1035 4      END;
; 1036 4
; 1037 4      CODECOMMENT
; 1038 4      '
; 1039 4      'Now put a unique pattern in a different memory location, and then',
; 1040 4      'set up for a DATI.',
; 1041 4      '
; 1042 5      BEGIN
; 1043 5
; 1044 5      SCRATCH[1] = %X'12345678';
; 1045 5      UBUS_PF = .MAP_NUMBER;
; 1046 5      UBUS_BO = SCRATCH[1];
; 1047 5      UET_ADDR_REG = .UBUS_ADDR;
; 1048 5
; 1049 5      TEMP3 = .MAP_NUMBER<7,2> * 8;
; 1050 5
; 1051 4      END;
; 1052 4
; 1053 4      CODECOMMENT
; 1054 4      '
; 1055 4      'Set up the UBI map. Put the CMI page frame nuber into the',
; 1056 4      'register BUF_PFN. Then call the map set up routine with the',
; 1057 4      'parameters map number, PFN of the CMI buffer, no byte offset,',
; 1058 4      'and data path number.',
; 1059 4      '
; 1060 5      BEGIN

```

```

; 1061 5
; 1062 5      TEMP = SCRATCH[1];
; 1063 5      BUF_PFN = .TEMP<9,15>;
; 1064 5      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.N);
; 1065 5
; 1066 4      END;
; 1067 4
; 1068 4      CODECOMMENT
; 1069 4      '
; 1070 4      'Execute the DATI, and after a nominal wait, check that the',
; 1071 4      'autopurge of the BDP occurred.',
; 1072 4      '
; 1073 5      BEGIN
; 1074 5
; 1075 5          UET_CTRL_REG = .TEMP3 OR DATI;
; 1076 5
; 1077 5          $DS_WAITUS(TIME=10);
; L 1078 5          UBI_STATUS(.N,UBIMOD_BDP);          ! M7
; %PRINT:      Test 18, Subtest 0, Error 4
; L 1079 6          UET_STATUS(UBIMOD_BDP);          ! M7
; %PRINT:      Test 18, Subtest 0, Error 5
; 1080 5          IF .SCRATCH[0] NEQ %X'00001234'
; 1081 5          THEN
; 1082 6              BEGIN
; P 1083 6                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
; P 1084 6                      PRLINK = PRINT_GENERAL,
; P 1085 6                      P1 = FMT_BDP_DATI,P2 = 3,
; L 1086 6                      P3 = .N,P4 = %X'00001234',P5 = .SCRATCH[0]);
; %PRINT:      Test 18, Subtest 0, Error 6
; 1087 5          END;
; 1088 5
; 1089 4      END;
; 1090 4
; 1091 4      CODECOMMENT
; 1092 4      '
; 1093 4      'Now check that the DATI worked.',
; 1094 4      '
; 1095 5      BEGIN
; 1096 5
; 1097 5          TEMP1 = .UET_DATA_REG;
; 1098 5          IF .TEMP1 NEQ %X'5678'
; 1099 5          THEN
; 1100 6              BEGIN
; P 1101 6                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
; P 1102 6                      PRLINK = PRINT_GENERAL,
; P 1103 6                      P1 = FMT_BDP_DATI,P2 = 3,
; L 1104 6                      P3 = .N,P4 = %X'5678',P5 = .TEMP1);
; %PRINT:      Test 18, Subtest 0, Error 7
; 1105 5          END;
; 1106 5
; 1107 4      END;
; 1108 4
; 1109 4      END
; 1110 3      WHILE $DS_INLOOP;          ! Scope loop ends here
; 1111 3

```


22-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_018: Buffered Data Path Autopurge Test
TEST_018: Buffered Data Path Autopurge Test

N 11

6-Sep-1985

Fiche 2

Frame N11

Sequence 349

6-Sep-1985 17:20:57

VAX-11 Bliss-32 V4.1-746

Page 39

6-Sep-1985 17:15:01

[LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5

(5)

; 1112 2
; 1113 2
; 1114 1
END;
\$DS_ENDTEST;

00000000' 00000000' 00000000' 00000000' 00030 \$:
00000000 00000003 00040

.PSECT DISPATCH,NOWRT,NOEXE,2

.ADDRESS P.AAG, P.AAH, P.AAI, TEST_018
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

20 61 74 61 44 20 64 65 72 65 66 66 75 42 21 0012 00048 P.AAG: .WORD 18
20 65 67 72 75 70 6F 74 75 41 20 68 74 61 50 0004A P.AAH: .ASCII \!Buffered Data Path Autopurge Test\
74 73 65 54 00059
00000000 0006C P.AAI: .LONG 0

.PSECT TEST_018,NOWRT,2

OFFC 00000

.ENTRY TEST_018, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0899
R11

5B 0000G CF 9E 00002
5A 0000' CF 9E 00007
59 00000000G 9F 9E 0000C

MOVAB SCRATCH, R11
MOVAB STATUS0, R10
MOVAB a#DS\$ERRHARD, R9

; Call the Unibus sizer routine to locate any Unibus memory. T
his builds a
; table of useable map entries. In the test which follows, the
map selected
; is drawn from this table.

0000G CF 00 FB 00013
58 01 D0 00018

CALLS #0, UNIBUS_SIZER
MOVL #1, N

; 0949
; 0952

; Purge the BDP. Clear a longword aligned location in memory.
Put a
; unique pattern into the UET data register.

0000GDF48 01 D0 0001B
7E 0A 7D 00021
00000000G 9F 02 FB 00024
6B D4 0002B

1\$: MOVL #1, aUBI_UNDER_TEST[N]
MOVQ #10, -(SP)
CALLS #2, a#DS\$WAITUS
CLRL SCRATCH

; 0965
; 0966
; 0967

; Set up the UET address register with the Unibus address. Bit
S
; 9 to 15 (the Unibus page frame) contain the 7 LSBs of the map
; number. Bits 0 to 8 contain the byte number field of the CMI
; address.

50 0000G CF 0003F462 8F C9 0002D
60 1234 8F B0 00037

BISL3 #259170, UNIBUS_SPACE, R0
MOVW #4660, (R0)

; 0978
; 0980

```

57      07      53      0000GCF48 3C 0003C      MOVZWL FREE_MAP[N], MAP_NUMBER      ; 0981
      09      53 F0 00042      INSV MAP_NUMBER, #9, #7, UBUS_ADDR      ; 0982
      50      6B 9E 00047      MOVAB SCRATCH, R0      ; 0983
57      09      50 F0 0004A      INSV R0, #0, #9, UBUS_ADDR      ;
      50 0000G CF 0003F460 8F C9 0004F      BISL3 #259168, UNIBUS_SPACE, R0      ;
      60      57 B0 00059      MOVW UBUS_ADDR, (R0)      ; 0984

; Write the 2 MSBs of the Unibus address with the 2 MSBs of the
; map number.
;
56      53      02      07 EF 0005C      EXTZV #7, #2, MAP_NUMBER, TEMP3      ; 0995
      56      08 C4 00061      MULL2 #8, TEMP3      ;

; Set up the UBI map. Put the CMI page frame number (PFN) into
; the
; register BUF_PFN. Then call the map setup routine with the
; parameters map number, PFN of the CMI buffer, no byte offset,
; and
; data path number.
;
52      54      54      6B 9E 00064      MOVAB SCRATCH, TEMP      ; 1008
      09 EF 00067      EXTZV #9, #15, TEMP, BUF_PFN      ; 1009
      58 DD 0006C      PUSHL N      ; 1010
      7E D4 0006E      CLRL -(SP)      ;
      52 DD 00070      PUSHL BUF_PFN      ;
      53 DD 00072      PUSHL MAP_NUMBER      ;
      04 FB 00074      CALLS #4, MAP_SETUP      ;

; Execute a DAT0, and after a nominal wait, check that the memo
; ry
; location still contains zero.
;
      50 0000G CF 0003F464 8F C9 00079      BISL3 #259172, UNIBUS_SPACE, R0      ; 1019
      60      05 A9 00083      BISH3 #5, TEMP3, (R0)      ; 1021
      7E      0A 7D 00087      MOVQ #10, -(SP)      ; 1023
      04 AA 00000000G 9F      CALLS #2, @DS$WAITUS      ;
      0000GDF48 1FFFFFFE 8F CB 00091      BICL3 #536870910, @UBI_UNDER_TEST[N], STATUS1      ; 1024
      50      04 AA D0 0009D      MOVL STATUS1, R0      ;
      1C 18 000A1      BGEQ 2$      ;
      7E 7C 000A3      CLRQ -(SP)      ;
      7E D4 000A5      CLRL -(SP)      ;
      50 DD 000A7      PUSHL R0      ;
      7E D4 000A9      CLRL -(SP)      ;
      58 DD 000AB      PUSHL N      ;
      0000G CF 9F 000AD      PUSHAB PRINT_CSR_ERR      ;
      0000G CF 9F 000B1      PUSHAB UBIMOD_BDP      ;
      7E 0000G CF 9A 000B5      MOVZBL LUN, -(SP)      ;
      01 DD 000BA      PUSHL #1      ;
      0A FB 000BC      CALLS #10, DS$ERRHARD      ;
      50 0000G CF 0003F464 8F C9 000BF 2$: BISL3 #259172, UNIBUS_SPACE, R0      ; 1025
      6A      60 B0 000C9      MOVW (R0), STATUS0      ;
      6A FF1E 8F AA 000CC      BICW2 #65310, STATUS0      ;
      50      6A 32 000D1      CVTWL STATUS0, R0      ;
      1D 13 000D4      BEQL 3$      ;
      7E 7C 000D6      CLRQ -(SP)      ;

```

```

50 DD 000D8
7E 02 7D 000DA
0000G CF 9F 000DD
0000G CF 9F 000E1
0000G CF 9F 000E5
7E 0000G CF 9A 000E9
02 DD 000EE
69 0A FB 000F0
50 6B D0 000F3
20 13 000F6
7E D4 000F8
50 DD 000FA
7E D4 000FC
58 DD 000FE
03 DD 00100
0000G CF 9F 00102
0000G CF 9F 00106
0000G CF 9F 0010A
7E 0000G CF 9A 0010E
03 DD 00113
69 0A FB 00115

```

```

PUSHL R0
MOVQ #2, -(SP)
PUSHAB FMT_UET_STAT
PUSHAB PRINT_GENERAL
PUSHAB UBIMOD_BDP
MOVZBL LUN, -(SP)
PUSHL #2
CALLS #10, DS$ERRHARD
MOVL SCRATCH, R0
BEQL 4$
CLRL -(SP)
PUSHL R0
CLRL -(SP)
PUSHL N
PUSHL #3
PUSHAB FMT_BDP_DATO
PUSHAB PRINT_GENERAL
PUSHAB UBIMOD_BDP
MOVZBL LUN, -(SP)
PUSHL #3
CALLS #10, DS$ERRHARD

```

1026

1032

```

;
; Now put a unique pattern in a different memory location, and
; then
; set up for a DATI.
;

```

```

57 07 04 AB 12345678 8F D0 00118
09 53 F0 00120
50 04 AB 9E 00125
57 09 00 50 F0 00129
50 0000G CF 0003F460 8F C9 0012E
60 57 B0 00138
56 53 02 07 EF 0013B
56 08 C4 00140

```

```

4$: MOVL #305419896, SCRATCH+4
INSV MAP_NUMBER, #9, #7, UBUS_ADDR
MOVAB SCRATCH+4, R0
INSV R0, #0, #9, UBUS_ADDR
BISL3 #259168, UNIBUS_SPACE, R0
MOVW UBUS_ADDR, (R0)
EXTZV #7, #2, MAP_NUMBER, TEMP3
MULL2 #8, TEMP3

```

1044

1045

1046

1047

1049

```

;
; Set up the UBI map. Put the CMI page frame nuber into the
; register BUF_PFN. Then call the map set up routine with the
; parameters map number, PFN of the CMI buffer, no byte offset,
; and data path number.
;

```

```

52 54 54 04 AB 9E 00143
0F 09 EF 00147
58 DD 0014C
7E D4 0014E
52 DD 00150
53 DD 00152
0000G CF 04 FB 00154

```

```

MOVAB SCRATCH+4, TEMP
EXTZV #9, #15, TEMP, BUF_PFN
PUSHL N
CLRL -(SP)
PUSHL BUF_PFN
PUSHL MAP_NUMBER
CALLS #4, MAP_SETUP

```

1062

1063

1064

```

;
; Execute the DATI, and after a nominal wait, check that the
; autopurge of the BDP occurred.
;

```

```

50 0000G CF 0003F464 8F C9 00159
60 56 01 A9 00163
7E 0A 7D 00167
00000000G 9F 02 FB 0016A

```

```

BISL3 #259172, UNIBUS_SPACE, R0
BISW3 #1, TEMP3, (R0)
MOVQ #10, -(SP)
CALLS #2, a#DS$WAITUS

```

1073

1075

1077

PC	Op	OpC	OpD	OpI	OpJ	OpK	OpL	OpM	OpN	OpO	OpP	OpQ	OpR	OpS	OpT	OpU	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN	OpSO
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

ZZ-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_018: Buffered Data Path Autopurge Test
TEST_018: Buffered Data Path Autopurge Test

E 12

6-Sep-1985

Fiche 2

Frame E12

Sequence 353

6-Sep-1985 17:20:57

VAX-11 Bliss-32 V4.1-746

Page 43

6-Sep-1985 17:15:01

[LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5

(5)

```

      7E      0000G  CF  9A 0022D
                        07 DD 00232
      69      0A FB 00234
00000000G  9F      00 FB 00237 8$:
      03      50 E9 0023E
                        FDD7 31 00241 9$:
F9      58      03 F3 00244 10$:
00000000G  9F      00 FB 00248
      50      01 D0 0024F
                        04 00252
```

```

MOVZBL  LUN, -(SP)
PUSHL   #7
CALLS   #10, DS$ERRHARD
CALLS   #0, a#DS$INLOOP
BLBC    R0, 10$
BRW     1$
AOBLEQ  #3, N, 9$
CALLS   #0, a#DS$BREAK
MOVL    #1, R0
RET
```

```

;
;
;
; 1110
;
;
; 0952
; 1112
; 1114
;
```

; Routine Size: 595 bytes, Routine Base: TEST_018 + 0000

```
; 1115 2 $DS_BGNTTEST (SECTION=(DEFAULT,ALL),TEXT='Byte Offset DATI Test');
; 1116 2
; 1117 2 !++
; 1118 2 ! TEST DESCRIPTION:
; 1119 2 !
; 1120 2 !     This tests the ability of the UBI to perform a DATI transaction
; 1121 2 !     in byte offset mode. All four data paths are used.
; 1122 2 !
; 1123 2 !     Before testing starts, the Unibus Sizer routine is called to determine
; 1124 2 !     which maps are useable.
; 1125 2 !
; 1126 2 ! ASSUMPTIONS:
; 1127 2 !
; 1128 2 !     Test 1-Test 7
; 1129 2 ! --
; 1130 2
; 1131 2 REGISTER
; 1132 2     BUF_PFN,
; 1133 2     MAP_NUMBER,
; 1134 2     TEMP,
; 1135 2     TEMP1:WORD,
; 1136 2     TEMP3,
; 1137 2     UBUS_ADDR: WORD;
; 1138 2
; 1139 2 MACRO
; 1140 2     UBUS_PF = UBUS_ADDR<9,7>x,      ! Defines the Unibus page frame field
; 1141 2     UBUS_BO = UBUS_ADDR<0,9>x;      ! Defines the Unibus byte number field
; 1142 2
; 1143 2 MAP
; 1144 2     SCRATCH: VECTOR[5,WORD];
; 1145 2
; 1146 2 CODECOMMENT
; 1147 2 ''
; 1148 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 1149 2 'a table of useable map entries. In the test which follows, the map',
; 1150 2 'selected is drawn from this table.',
; 1151 2 '';
; 1152 3     BEGIN
; 1153 3
; 1154 3     UNIBUS_SIZER();
; 1155 3
; 1156 2     END;
; 1157 2
; 1158 2 CODECOMMENT
; 1159 2 ''
; 1160 2 'Set up the UBI UET for a DATI, and set',
; 1161 2 'up the CMI source location (longword aligned for M = 0, word aligned',
; 1162 2 'for M = 1) with the appropriate data pattern. Execute the DATI and',
; 1163 2 'check that the UET data register contains the data pattern.',
; 1164 2 'Repeat 10 times, once for each of the five data patterns with the two',
; 1165 2 'variations of alignment.',
; 1166 2 '';
; 1167 3     BEGIN
; 1168 3
; 1169 3     INCR I FROM 0 TO 3 DO
```



```

; 1170 4      BEGIN
; 1171 4
; 1172 4      INCR M FROM 0 TO 1 DO
; 1173 5          BEGIN
; 1174 5
; 1175 5      DO                                ! Scope loop begins here
; 1176 6          BEGIN
; 1177 6
; 1178 6      IF .I GTR 0
; 1179 6      THEN
; 1180 6          UBI_CSR(.I) = PURGE;
; 1181 6
; 1182 6      CODECOMMENT
; 1183 6      ''
; 1184 6      'Set up the UET address register with the Unibus',
; 1185 6      'address. Bits 9 to 15 (the Unibus page frame) contain',
; 1186 6      'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
; 1187 6      'the byte number field of the CMI address.',
; 1188 6      ''
; 1189 7          BEGIN
; 1190 7
; 1191 7          SCRATCH[.M] + 1 = %X'12345678';
; 1192 7          UET_DATA_REG = 0;
; 1193 7          MAP_NUMBER = .FREE_MAP[.I];
; 1194 7          UBUS_PF = .MAP_NUMBER;
; 1195 7          UBUS_BO = SCRATCH[.M];
; 1196 7          UET_ADDR_REG = .UBUS_ADDR;
; 1197 7
; 1198 6      END;
; 1199 6
; 1200 6      CODECOMMENT
; 1201 6      ''
; 1202 6      'Write the 2 MSBs of the Unibus address with the 2 MSBs of',
; 1203 6      'the map number.',
; 1204 6      ''
; 1205 7          BEGIN
; 1206 7
; 1207 7          TEMP3 = .MAP_NUMBER<7,2> * 8;
; 1208 7
; 1209 6      END;
; 1210 6
; 1211 6      CODECOMMENT
; 1212 6      ''
; 1213 6      'Set up the UBI map. Put the CMI page frame number (PFN)',
; 1214 6      'into the register BUF_PFN. Then call the map setup routine',
; 1215 6      'with the parameters map number, PFN of the CMI buffer,',
; 1216 6      'address offset enabled, and data path number.',
; 1217 6      ''
; 1218 7          BEGIN
; 1219 7
; 1220 7          TEMP = SCRATCH[.M];
; 1221 7          BUF_PFN = .TEMP<9,15>;
; 1222 7          MAP_SETUP(.MAP_NUMBER,.BUF_PFN,1,.I);
; 1223 7
; 1224 6      END;

```

```

; 1225 6
; 1226 6
; 1227 6
; 1228 6
; 1229 6
; 1230 6
; 1231 7
; 1232 7
; 1233 7
; 1234 7
; 1235 7
; L 1236 7
; xPRINT: Test 19, Subtest 0, Error 1
; L 1237 8
; xPRINT: Test 19, Subtest 0, Error 2
; 1238 7
; 1239 7
; 1240 7
; 1241 8
; P 1242 8
; P 1243 8
; P 1244 8
; L 1245 8
; xPRINT: Test 19, Subtest 0, Error 3
; 1246 7
; 1247 7
; 1248 6
; 1249 6
; 1250 6
; 1251 6
; 1252 6
; 1253 6
; 1254 6
; 1255 6
; 1256 7
; 1257 7
; 1258 7
; 1259 7
; 1260 7
; 1261 7
; 1262 7
; L 1263 7
; xPRINT: Test 19, Subtest 0, Error 4
; L 1264 8
; xPRINT: Test 19, Subtest 0, Error 5
; 1265 7
; 1266 7
; 1267 7
; 1268 8
; P 1269 8
; P 1270 8
; P 1271 8
; L 1272 8
; xPRINT: Test 19, Subtest 0, Error 6
; 1273 7

```

```

CODECOMMENT
''
'Execute the DATI and after a nominal wait check that the',
'UET data register contains the expected data.',
''
BEGIN
    UET_CTRL_REG = .TEMP3 OR DATI;

    $DS_WAITUS(TIME=10);
    IF .I GTR 0 THEN UBI_STATUS(.I,UBIMOD_BDP); ! M7
    UET_STATUS(UBIMOD_BDP); ! M7
    TEMP1 = .UET_DATA_REG;
    IF .TEMP1 NEQ 'X'5678'
    THEN
        BEGIN
            $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                PRLINK = PRINT_GENERAL,
                P1 = FMT_BDP_DATI,P2 = 3,
                P3 = .I,P4 = 'X'5678',P5 = .TEMP1);
        END;
    END;

CODECOMMENT
''
'Change Unibus address bit one from a zero to a one and',
'executed another DATI. After a nominal wait, check that',
'the data in the UET data register is correct.',
''
BEGIN
    UET_ADDR_REG = .UBUS_ADDR + 2;
    UET_CTRL_REG = .TEMP3 OR DATI;

    $DS_WAITUS(TIME=10);
    IF .I GTR 0 THEN UBI_STATUS(.I,UBIMOD_BDP); ! M7
    UET_STATUS(UBIMOD_BDP); ! M7
    TEMP1 = .UET_DATA_REG;
    IF .TEMP1 NEQ 'X'1234'
    THEN
        BEGIN
            $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                PRLINK = PRINT_GENERAL,
                P1 = FMT_BDP_DATI,P2 = 3,
                P3 = .I,P4 = 'X'1234',P5 = .TEMP1);
        END;
    END;

```

```
; 1274 7
; 1275 6
; 1276 6
; 1277 6
; 1278 5
; 1279 5
; 1280 4
; 1281 4
; 1282 3
; 1283 3
; 1284 2
; 1285 2
; 1286 1

                END;
                END
        WHILE $DS_INLOOP;    ! Scope loop ends here
        END;
        END;
        END;
        $DS_ENDTEST;
```

```
                                .PSECT DISPATCH,NOWRT,NOEXE,2
                                .ADDRESS P.AAJ, P.AAK, P.AAL, TEST_019
                                .LONG 3, 0
                                .PSECT $PLIT$,NOWRT,NOEXE,2
00000000' 00000000' 00000000' 00000000' 00048 $:
                                00000000 00000003 00058
                                .PSECT TEST_019,NOWRT,2
                                .ENTRY TEST_019, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1115
                                MOVAB STATUS0, R11
                                MCVAB @DS$ERRHARD, R10
; Call the Unibus sizer routine to locate any Unibus memory. T
; his builds
; a table of useable map entries. In the test which follows, t
; he map
; selected is drawn from this table.
                                CALLS #0, UNIBUS_SIZER ; 1154
; Set up the UBI UET for a DATI, and set
; up the CMI source location (longword aligned for M = 0, word
; aligned
; for M = 1) with the appropriate data pattern. Execute the DAT
; I and
; check that the UET data register contains the data pattern.
; Repeat 10 times, once for each of the five data patterns with
; the two
; variations of alignment.
                                0000G CF 00 FB 0000E
```

```
41 44 20 74 65 73 66 66 4F 20 65 74 79 42 15 0013 00070 P.AAJ: .WORD 19
74 73 65 54 20 49 54 00072 P.AAK: .ASCII <21>\Byte Offset DATI Test\
00000000 00088 P.AAL: .LONG 0
```



```

58 D4 00013 CLRL I ; 1169
59 D4 00015 1$: CLRL M ; 1172
58 D5 00017 2$: TSTL I ; 1178
06 15 00019 BLEQ 3$ ;
01 D0 0001B MOVL #1, aUBI_UNDER_TEST[I] ; 1180
;
; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.
;
0000GCF49 3F 00021 3$: PUSHAW SCRATCH+1[M] ; 1191
50 0000G 9E 12345678 8F D0 00026 MOVL #305419896, a(SP)+ ;
CF 0003F462 8F C9 0002D BISL3 #259170, UNIBUS_SPACE, R0 ;
60 B4 00037 CLRW (R0) ; 1192
57 07 53 0000GCF48 3C 00039 MOVZWL FREE_MAP[I], MAP_NUMBER ; 1193
09 53 F0 0003F INSV MAP_NUMBER, #9, #7, UBUS_ADDR ; 1194
57 09 50 0000GCF49 3E 00044 MOVAW SCRATCH[M], R0 ; 1195
00 50 F0 0004A INSV R0, #0, #9, UBUS_ADDR ;
50 0000G CF 0003F460 8F C9 0004F BISL3 #259168, UNIBUS_SPACE, R0 ;
60 57 B0 00059 MOVW UBUS_ADDR, (R0) ; 1196
;
; Write the 2 MSBs of the Unibus address with the 2 MSBs of
; the map number.
;
56 53 02 07 EF 0005C EXTZV #7, #2, MAP_NUMBER, TEMP3 ; 1207
56 08 C4 00061 MULL2 #8, TEMP3 ;
;
; Set up the UBI map. Put the CMI page frame number (PFN)
; into the register BUF_PFN. Then call the map setup routine
; with the parameters map number, PFN of the CMI buffer,
; address offset enabled, and data path number.
;
52 54 54 0000GCF49 3E 00064 MOVAW SCRATCH[M], TEMP ; 1220
0F 09 EF 0006A EXTZV #9, #15, TEMP, BUF_PFN ; 1221
58 DD 0006F PUSHL I ; 1222
01 DD 00071 PUSHL #1 ;
52 DD 00073 PUSHL BUF_PFN ;
53 DD 00075 PUSHL MAP_NUMBER ;
0000G CF 04 FB 00077 CALLS #4, MAP_SETUP ;
;
; Execute the DATI and after a nominal wait check that the
; UET data register contains the expected data.
;
50 0000G CF 0003F464 8F C9 0007C BISL3 #259172, UNIBUS_SPACE, R0 ; 1231
60 56 01 A9 00086 BISW3 #1, TEMP3, (R0) ; 1233
7E 0A 7D 0008A MOVQ #10, -(SP) ; 1235
00000000G 9F 02 FB 0008D CALLS #2, a#DS$WAITUS ;
58 D5 00094 TSTL I ; 1236
0C 15 00096 BLEQ 4$ ;
04 AB 0000GDF48 1FFFFFFE 8F CB 00098 BICL3 #536870910, aUBI_UNDER_TEST[I], STATUS1 ;
50 94 AB D0 000A4 4$: MOVL STATUS1, R0 ;
1C 18 000A8 BGEQ 5$ ;
7E 7C 000AA CLRQ -(SP) ;
7E D4 000AC CLRL -(SP) ;

```

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

			50	DD	00172	PUSHL	R0	:	
			7E	D4	00174	CLRL	-(SP)	:	
			58	DD	00176	PUSHL	I	:	
		0000G	CF	9F	00178	PUSHAB	PRINT_CSR_ERR	:	
		0000G	CF	9F	0017C	PUSHAB	UBIMOD_BDP	:	
	7E	0000G	CF	9A	00180	MOVZBL	LUN, -(SP)	:	
			04	DD	00185	PUSHL	#4	:	
	6A		0A	FB	00187	CALLS	#10, DS\$ERRHARD	:	
50	0000G	CF	0003F464	8F	C9	0018A	9\$: BISL3	#259172, UNIBUS_SPACE, R0	1264
	6B			60	B0	00194	MOVW	(R0), STATUS0	:
	6B	FF1E		8F	AA	00197	BICW2	#65310, STATUS0	:
	50			6B	32	0019C	CVTWL	STATUS0, R0	:
				1D	13	0019F	BEQL	10\$	:
				7E	7C	001A1	CLRQ	-(SP)	:
				50	DD	001A3	PUSHL	R0	:
	7E			02	7D	001A5	MOVQ	#2, -(SP)	:
		0000G		CF	9F	001A8	PUSHAB	FMT_UET_STAT	:
		0000G		CF	9F	001AC	PUSHAB	PRINT_GENERAL	:
		0000G		CF	9F	001B0	PUSHAB	UBIMOD_BDP	:
	7E	0000G		CF	9A	001B4	MOVZBL	LUN, -(SP)	:
				05	DD	001B9	PUSHL	#5	:
	6A			0A	FB	001BB	CALLS	#10, DS\$ERRHARD	:
50	0000G	CF	0003F462	8F	C9	001BE	10\$: BISL3	#259170, UNIBUS_SPACE, R0	1265
	55			60	B0	001C8	MOVW	(R0), TEMP1	:
	1234	8F		55	B1	001CB	CMPW	TEMP1, #4660	1266
				24	13	001D0	BEQL	11\$	:
				7E	D4	001D2	CLRL	-(SP)	1272
	7E			55	3C	001D4	MOVZWL	TEMP1, -(SP)	:
	7E	1234		8F	3C	001D7	MOVZWL	#4660, -(SP)	:
				58	DD	001DC	PUSHL	I	:
				03	DD	001DE	PUSHL	#3	:
		0000G		CF	9F	001E0	PUSHAB	FMT_BDP_DATI	:
		0000G		CF	9F	001E4	PUSHAB	PRINT_GENERAL	:
		0000G		CF	9F	001E8	PUSHAB	UBIMOD_BDP	:
	7E	0000G		CF	9A	001EC	MOVZBL	LUN, -(SP)	:
				06	DD	001F1	PUSHL	#6	:
	6A			0A	FB	001F3	CALLS	#10, DS\$ERRHARD	:
	00000000G	9F		00	FB	001F6	11\$: CALLS	#0, a#DS\$INLOOP	1278
		03		50	E9	001FD	BLBC	R0, 13\$	:
			FE14	31	00200	12\$: BRW	2\$	:	
	F9	59		01	F3	00203	13\$: AOBLEQ	#1, M, 12\$	1172
FE08	58	01		03	F1	00207	ACBL	#3, #1, I, 1\$	1169
	00000000G	9F		00	FB	0020D	CALLS	#0, a#DS\$BREAK	1284
		50		01	D0	00214	MOVL	#1, R0	1286
				04	00217	RET		:	

; Routine Size: 536 bytes, Routine Base: TEST_019 + 0000


```
; 1287 2 $DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Byte Offset DATIP/DATO Test');
; 1288 2
; 1289 2 !**
; 1290 2 ! TEST DESCRIPTION:
; 1291 2 !
; 1292 2 !     This tests the ability of the UBI to perform a DATIP/DATO transaction
; 1293 2 !     in address offset mode. Only the 3 Buffered Data Paths are tested.
; 1294 2 !
; 1295 2 !     Before testing starts, the Unibus Sizer routine is called to determine
; 1296 2 !     which maps are useable.
; 1297 2 !
; 1298 2 ! ASSUMPTIONS:
; 1299 2 !
; 1300 2 !     Test 1-Test 7
; 1301 2 !--
; 1302 2
; 1303 2 REGISTER
; 1304 2     BUF_PFN,
; 1305 2     MAP_NUMBER,
; 1306 2     TEMP,
; 1307 2     TEMP1,
; 1308 2     TEMP3,
; 1309 2     UBUS_ADDR: WORD;
; 1310 2
; 1311 2 BUILTIN
; 1312 2     MTPR;
; 1313 2
; 1314 2 MACRO
; 1315 2     UBUS_PF = UBUS_ADDR<9,7>x,      ! Defines the Unibus page frame field
; 1316 2     UBUS_BO = UBUS_ADDR<0,9>x;      ! Defines the Unibus byte number field
; 1317 2
; 1318 2 MAP
; 1319 2     SCRATCH: VECTOR[5,WORD];
; 1320 2
; 1321 2 CODECOMMENT
; 1322 2 ''
; 1323 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 1324 2 'a table of useable map entries. In the test which follows, the map',
; 1325 2 'selected is drawn from this table.',
; 1326 2 ''
; 1327 3     BEGIN
; 1328 3
; 1329 3     UNIBUS_SIZER();
; 1330 3
; 1331 2     END;
; 1332 2
; 1333 2 CODECOMMENT
; 1334 2 ''
; 1335 2 'Set up the UBI UET for a DATI, and set',
; 1336 2 'up the CMI source location (longword aligned for M = 0, word aligned',
; 1337 2 'for M = 1) with the appropriate data pattern. Execute the DATI and',
; 1338 2 'check that the UET data register contains the data pattern.',
; 1339 2 'Repeat 6 times, once for each of the three data paths with the two',
; 1340 2 'variations of alignment.',
; 1341 2 ''
```

```

: 1342 3      BEGIN
: 1343 3
: 1344 3      INCR I FROM 1 TO 3 DO
: 1345 4          BEGIN
: 1346 4
: 1347 4          INCR M FROM 0 TO 1 DO
: 1348 5              BEGIN
: 1349 5
: 1350 5          DO                      ! Scope loop begins here
: 1351 6              BEGIN
: 1352 6
: 1353 6          UBI_CSR(.I) = PURGE;
: 1354 6
: 1355 6      CODECOMMENT
: 1356 6      ''
: 1357 6      'Set up the UET address register with the Unibus',
: 1358 6      'address. Bits 9 to 15 (the Unibus page frame) contain',
: 1359 6      'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
: 1360 6      'the byte number field of the CMI address.',
: 1361 6      ''
: 1362 7          BEGIN
: 1363 7
: 1364 7          SCRATCH(.M) + 1 = %X'3333';
: 1365 7          UET_DATA_REG = 0;
: 1366 7          MAP_NUMBER = .FREE_MAP(.I);
: 1367 7          UBUS_PF = .MAP_NUMBER;
: 1368 7          UBUS_BO = SCRATCH(.M);
: 1369 7          UET_ADDR_REG = .UBUS_ADDR;
: 1370 7
: 1371 6          END;
: 1372 6
: 1373 6      CODECOMMENT
: 1374 6      ''
: 1375 6      'Write the 2 MSBs of the Unibus address with the 2 MSBs of',
: 1376 6      'the map number.',
: 1377 6      ''
: 1378 7          BEGIN
: 1379 7
: 1380 7          TEMP3 = .MAP_NUMBER<7,2> * 8;
: 1381 7
: 1382 6          END;
: 1383 6
: 1384 6      CODECOMMENT
: 1385 6      ''
: 1386 6      'Set up the UBI map. Put the CMI page frame number (PFN)',
: 1387 6      'into the register BUF_PFN. Then call the map setup routine',
: 1388 6      'with the parameters map number, PFN of the CMI buffer,',
: 1389 6      'address offset enabled, and data path number.',
: 1390 6      ''
: 1391 7          BEGIN
: 1392 7
: 1393 7          TEMP = SCRATCH(.M);
: 1394 7          BUF_PFN = .TEMP<9,15>;
: 1395 7          MAP_SETUP(.MAP_NUMBER,.BUF_PFN,1,.I);
: 1396 7

```

```

; 1397 6
; 1398 6
; 1399 6
; 1400 6
; 1401 6
; 1402 6
; 1403 6
; 1404 6
; 1405 6
; 1406 7
; 1407 7
; 1408 7
; 1409 7
; 1410 7
; L 1411 7
; xPRINT: Test 20, Subtest 0, Error 1
; L 1412 8
; xPRINT: Test 20, Subtest 0, Error 2
; 1413 7
; 1414 7
; 1415 7
; 1416 7
; 1417 7
; 1418 8
; P 1419 8
; P 1420 8
; P 1421 8
; L 1422 8
; xPRINT: Test 20, Subtest 0, Error 3
; 1423 7
; 1424 7
; 1425 6
; 1426 6
; 1427 6
; 1428 5
; 1429 5
; 1430 4
; 1431 4
; 1432 3
; 1433 3
; 1434 2
; 1435 2
; 1436 1

```

```

END;

CODECOMMENT
''
'Execute the DATIP and after a nominal wait check that the',
'UET data register contains the expected data',
'pattern. Check also that the DATO portion of the',
'transaction was successful.',
''
BEGIN
    UET_CTRL_REG = .TEMP3 OR DATIP;

    $DS_WAITUS(TIME=10);
    UBI_STATUS(.I,UBIMOD_DDP); ! M7
    UET_STATUS(UBIMOD_DDP); ! M7
    TEMP = .UET_DATA_REG;
    TEMP1 = -1;
    MTPR(TEMP1,x'37'); ! Cleanup the CMI ! A8
    IF .TEMP NEQ x'3333'
    THEN
        BEGIN
            $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
                PRLINK = PRINT_GENERAL,
                P1 = FMT_BDP_DATI,P2 = 3,
                P3 = .I,P4 = x'6666',P5 = .TEMP);
        END
    END
    WHILE $DS_INLOOP; ! Scope loop ends here
END;
END;
END;
$DS_ENDTEST;

```

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00060 \$:
00000000 00000003 00070

.ADDRESS P.AAM, P.AAN, P.AAO, TEST_020
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

41 44 20 74 65 73 66 66 4F 20 65 74 79 42 1B 0014 0008C P.AAM: .WORD 20
0008E P.AAN: .ASCII <27>\Byte Offset DATIP/DATO Test\

ZZ-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_020: Byte Offset DATIP/DATO Test
TEST_020: Byte Offset DATIP/DATO Test

C 13

6-Sep-1985 Fiche 2 Frame C13 Sequence 364
6-Sep-1985 17:20:57 VAX-11 Bliss-32 V4.1-746 Page 54
6-Sep-1985 17:15:01 [LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5 (7)

74 73 65 54 20 4F 54 41 44 2F 50 49 54 0009D
000AA
00000000 000AC P.AAO: .BLKB 2
.LONG 0

.PSECT TEST_020,NOWRT,2

OFFC 00000

.ENTRY TEST_020, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1287
R11

5B 00000000G 9F 9E 00002

MOVAB @DS\$ERRHARD, R11

; Call the Unibus sizer routine to locate any Unibus memory. T
his builds
; a table of useable map entries. In the test which follows, t
he map
; selected is drawn from this table.

0000G CF 00 FB 00009

CALLS #0, UNIBUS_SIZER

; 1329

; Set up the UBI UET for a DATI, and set
; up the CMI source location (longword aligned for M = 0, word
aligned
; for M = 1) with the appropriate data pattern. Execute the DAT
I and
; check that the UET data register contains the data pattern.
; Repeat 6 times, once for each of the three data paths with th
e two
; variations of alignment.

58 01 D0 0000E
59 D4 00011
5A 58 02 78 00013
5A 0000G CF C1 00017
60 01 D0 0001D

1\$: MOVL #1, I ; 1344
CLRL M ; 1347
2\$: ASHL #2, I, R10 ; 1353
3\$: ADDL3 UBI_UNDER_TEST, R10, R0
MOVL #1, (R0)

; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.

50 0000G 9E 0000GCF49 3F 00020
3333 8F 3C 00025
CF 0003F462 8F C9 0002A
60 B4 00034
53 0000GCF48 3C 00036
07 09 53 F0 0003C
50 0000GCF49 3E 00041
57 09 00 F0 00047
50 0000G CF 0003F460 8F C9 0004C
60 57 B0 00056

PUSHAW SCRATCH+1[M] ; 1364
MOVZWL #13107, @ (SP)+
BISL3 #259170, UNIBUS_SPACE, R0
CLRW (R0) ; 1365
MOVZWL FREE_MAP[I], MAP_NUMBER ; 1366
INSV MAP_NUMBER, #9, #7, UBUS_ADDR ; 1367
MOVAV SCRATCH[M], R0 ; 1368
INSV R0, #0, #9, UBUS_ADDR
BISL3 #259168, UNIBUS_SPACE, R0
MOVW UBUS_ADDR, (R0) ; 1369

; Write the 2 MSBs of the Unibus address with the 2 MSBs of
; the map number.

```

56          53          02          07 EF 00059      EXTZV  #7, #2, MAP_NUMBER, TEMP3      ; 1380
          56          08 C4 0005E      MULL2  #8, TEMP3      ;
;
; Set up the UBI map. Put the CMI page frame number (PFN)
; into the register BUF_PFN. Then call the map setup routine
; with the parameters map number, PFN of the CMI buffer,
; address offset enabled, and data path number.
;
52          54          54 0000GCF49 3E 00061      MOVAW  SCRATCH[M], TEMP      ; 1393
          0F          09 EF 00067      EXTZV  #9, #15, TEMP, BUF_PFN      ; 1394
          58          DD 0006C      PUSHL  I      ; 1395
          01          DD 0006E      PUSHL  #1      ;
          52          DD 00070      PUSHL  BUF_PFN      ;
          53          DD 00072      PUSHL  MAP_NUMBER      ;
          0000G CF          04 FB 00074      CALLS  #4, MAP_SETUP      ;
;
; Execute the DATIP and after a nominal wait check that the
; UET data register contains the expected data
; pattern. Check also that the DATO portion of the
; transaction was successful.
;
          50          0000G CF 0003F464 8F C9 00079      BISL3  #259172, UNIBUS_SPACE, R0      ; 1406
          60          56          03 A9 00083      BISW3  #3, TEMP3, (R0)      ; 1408
          7E          0A 7D 00087      MOVQ  #10, -(SP)      ; 1410
          00000000G 9F          02 FB 0008A      CALLS  #2, @DS$WAITUS      ;
          5A          58          02 78 00091      ASHL  #2, I, R10      ; 1411
          50          5A          0000G CF C1 00095      ADDL3  UBI_UNDER_TEST, R10, R0      ;
          0000' CF          60 1FFFFFFE 8F CB 0009B      BICL3  #536870910, (R0), STATUS1      ;
          50          0000' CF D0 000A5      MOVL  STATUS1, R0      ;
          1C 18 000AA      BGEQ  4$      ;
          7E 7C 000AC      CLRQ  -(SP)      ;
          7E D4 000AE      CLRL  -(SP)      ;
          50 DD 000B0      PUSHL  R0      ;
          7E D4 000B2      CLRL  -(SP)      ;
          58 DD 000B4      PUSHL  I      ;
          0000G CF 9F 000B6      PUSHAB PRINT_CSR_ERR      ;
          0000G CF 9F 000BA      PUSHAB UBIMOD_DDP      ;
          7E 0000G CF 9A 000BE      MOVZBL LUN, -(SP)      ;
          01 DD 000C3      PUSHL  #1      ;
          6B          0A FB 000C5      CALLS  #10, DS$ERRHARD      ;
          50          0000G CF 0003F464 8F C9 000C8 4$: BISL3  #259172, UNIBUS_SPACE, R0      ; 1412
          0000' CF          60 B0 000D2      MOVW  (R0), STATUS0      ;
          0000' CF          8F AA 000D7      BICW2  #65310, STATUS0      ;
          50          0000' CF 32 000DE      CVTWL  STATUS0, R0      ;
          1D 13 000E3      BEQL  5$      ;
          7E 7C 000E5      CLRQ  -(SP)      ;
          50 DD 000E7      PUSHL  R0      ;
          7E          02 7D 000E9      MOVQ  #2, -(SP)      ;
          0000G CF 9F 000EC      PUSHAB FMT_UET_STAT      ;
          0000G CF 9F 000F0      PUSHAB PRINT_GENERAL      ;
          0000G CF 9F 000F4      PUSHAB UBIMOD_DDP      ;
          7E 0000G CF 9A 000F8      MOVZBL LUN, -(SP)      ;
          02 DD 000FD      PUSHL  #2      ;
          6B          0A FB 000FF      CALLS  #10, DS$ERRHARD      ;
          50          0000G CF 0003F462 8F C9 00102 5$: BISL3  #259170, UNIBUS_SPACE, R0      ; 1413

```

ZZ-ECCBA-1.4 TEST_020: Byte Offset DATIP/DATO Test
UBI_TESTS_16_22
01-04 TEST_020: Byte Offset DATIP/DATO Test

E 13

6-Sep-1985 Fiche 2 Frame E13 Sequence 366
6-Sep-1985 17:20:57 VAX-11 Bliss-32 V4.1-746 Page 56
6-Sep-1985 17:15:01 [LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5 (7)

		54		60	3C	0010C	MOVZWL	(R0), TEMP	:	
		55		01	CE	0010F	MNEGL	#1, TEMP1	:	1414
		37		55	DA	00112	MTPR	TEMP1, #55	:	1415
	00003333	8F		54	D1	00115	CMPL	TEMP, #13107	:	1416
				23	13	0011C	BEQL	6\$	:	
				7E	D4	0011E	CLRL	-(SP)	:	1422
				54	DD	00120	PUSHL	TEMP	:	
	7E		6666	8F	3C	00122	MOVZWL	#26214, -(SP)	:	
				58	DD	00127	PUSHL	I	:	
				03	DD	00129	PUSHL	#3	:	
			0000G	CF	9F	0012B	PUSHAB	FMT_BDP_DATI	:	
			0000G	CF	9F	0012F	PUSHAB	PRINT_GENERAL	:	
			0000G	CF	9F	00133	PUSHAB	UBIMOD_DDP	:	
	7E		0000G	CF	9A	00137	MOVZBL	LUN, -(SP)	:	
				03	DD	0013C	PUSHL	#3	:	
		6B		0A	FB	0013E	CALLS	#10, DS\$ERRHARD	:	
	00000000G	9F		00	FB	00141	CALLS	#0, a#DS\$INLOOP	:	1428
		03		50	E9	00148	BLBC	R0, 7\$	:	
				FEC9	31	0014B	BRW	3\$	:	
				01	F1	0014E	ACBL	#1, #1, M, 2\$	:	1347
FEBF	59	01		03	F1	00154	ACBL	#3, #1, I, 1\$	:	1344
FEB7	58	01		00	FB	0015A	CALLS	#0, a#DS\$BREAK	:	1434
		9F		01	D0	00161	MOVL	#1, R0	:	1436
	00000000G	50		04	00164	RET			:	

; Routine Size: 357 bytes, Routine Base: TEST_020 + 0000


```
; 1437 2 $DS_BGNTTEST (SECTION=(DEFAULT,ALL),TEXT='Byte Offset DDP DATO Test');
; 1438 2
; 1439 2 !++
; 1440 2 ! TEST DESCRIPTION:
; 1441 2 !
; 1442 2 !     This test verifies the ability of the UBI to handle DATO transactions
; 1443 2 !     through the DDP with byte offset mode enabled. The behavior of the
; 1444 2 !     UBI in this mode is similar to its behavior in a normal DATO, with the
; 1445 2 !     exception that odd byte addressing of the CMI is accomplished in this
; 1446 2 !     mode.
; 1447 2 !
; 1448 2 ! ASSUMPTIONS:
; 1449 2 !
; 1450 2 !     Test 1-Test 15
; 1451 2 !--
; 1452 2
; 1453 2 REGISTER
; 1454 2     BUF_PFN,
; 1455 2     M,
; 1456 2     MAP_NUMBER,
; 1457 2     TEMP,
; 1458 2     TEMP0: WORD,
; 1459 2     TEMP3,
; 1460 2     UBUS_ADDR: WORD;
; 1461 2
; 1462 2 LOCAL
; 1463 2     TEMP4: WORD;                ! A10 M11
; 1464 2     ! TEMP5: VECTOR[2,WORD];    ! A10 D11
; 1465 2
; 1466 2 MACRO
; 1467 2     UBUS_PF = UBUS_ADDR<9,7>x,    ! Defines the Unibus page frame field
; 1468 2     UBUS_BO = UBUS_ADDR<0,9>x;    ! Defines the Unibus byte number field
; 1469 2
; 1470 2 MAP
; 1471 2     SCRATCH: VECTOR[3,WORD];
; 1472 2
; 1473 2 !TEMP5[0] = xX'3400';            ! A10 D11
; 1474 2 !TEMP5[1] = xX'3412';            ! A10 D11
; 1475 2
; 1476 2 CODECOMMENT
; 1477 2 ''
; 1478 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 1479 2 'a table of useable map entries. In the test which follows, the map',
; 1480 2 'selected is drawn from this table.',
; 1481 2 '';
; 1482 3     BEGIN
; 1483 3
; 1484 3     UNIBUS_SIZER();
; 1485 3     MAP_NUMBER = .FREE_MAP[0];
; 1486 3
; 1487 2     END;
; 1488 2
; 1489 2 CODECOMMENT
; 1490 2 ''
; 1491 2 'Set up the UBI and UET for a DATO on a longword alignment and word',
```

```
; 1492 2 'alignment. The first DATO should write the data pattern into SCRATCH + 1.',
; 1493 2 'and the second DATO will write the data pattern into SCRATCH + 3. The',
; 1494 2 'latter DATO will wrap across a longword boundary, and thus cause the UBI',
; 1495 2 'to perform two writes to memory.',
; 1496 2 '':
; 1497 3 BEGIN
; 1498 3
; 1499 3 INCR M FROM 0 TO 1 DO
; 1500 4 BEGIN
; 1501 4
; 1502 4 DO ! Scope loop begins here
; 1503 5 BEGIN
; 1504 5
; 1505 5 CODECOMMENT
; 1506 5 'Set up the UET address register with the Unibus address.',
; 1507 5 'Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs',
; 1508 5 'of the UBE map number. Bits 0 to 8 contain the byte number',
; 1509 5 'field of the CMI address.',
; 1510 5 '':
; 1511 5 BEGIN
; 1512 6
; 1513 6 SCRATCH[0] = 0; ! Clear the memory destinations
; 1514 6 SCRATCH[1] = 0;
; 1515 6 SCRATCH[2] = 0;
; 1516 6 UET_DATA_REG = X'1234';
; 1517 6 UBUS_PF = .MAP_NUMBER;
; 1518 6 UBUS_BO = SCRATCH[M];
; 1519 6 UET_ADDR_REG = .UBUS_ADDR;
; 1520 6
; 1521 6 END;
; 1522 5
; 1523 5 CODECOMMENT
; 1524 5 'Write the two MSBs of the Unibus address with the two MSBs of',
; 1525 5 'the map number.',
; 1526 5 '':
; 1527 5 BEGIN
; 1528 6
; 1529 6 TEMP3 = .MAP_NUMBER<7,2> * 8;
; 1530 6
; 1531 6 END;
; 1532 5
; 1533 5 CODECOMMENT
; 1534 5 'Set up the UBI map. Put the CMI page frame number (PFN)',
; 1535 5 'into the register BUF_PFN. Then call the map setup routine',
; 1536 5 'with the parameters map number, PFN of the CMI buffer, byte',
; 1537 5 'offset mode (1 = byte offset mode enabled), and data path',
; 1538 5 'number (0 for DDP).',
; 1539 5 '':
; 1540 5 BEGIN
; 1541 6
; 1542 6 TEMP = SCRATCH[M]; ! Get the address of SCRATCH
; 1543 6 BUF_PFN = .TEMP<9,15>;
; 1544 6
; 1545 6
; 1546 6
```

```

; 1547 6      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,1,0);
; 1548 6
; 1549 5      END;
; 1550 5
; 1551 5      CODECOMMENT
; 1552 5      '
; 1553 5      'Execute the DATO and after a nominal wait check that the',
; 1554 5      'CMI destination contains the correct data. If not, report',
; 1555 5      'the error.',
; 1556 5      '
; 1557 6      BEGIN
; 1558 6
; 1559 6      UET_CTRL_REG = .TEMP3 OR DATO;
; 1560 6
; 1561 6      $DS_WAITUS(TIME=10);
; L 1562 7      UET_STATUS(UBIMOD_DDP);          ! M7
; xPRINT:      Test 21, Subtest 0, Error 1
; 1563 6      IF (TEMP4 = .(SCRATCH[.M])<0,16>) NEQ xX'3400' ! M10 M11
; 1564 6      THEN
; 1565 7      BEGIN
; P 1566 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
; P 1567 7      PRLINK = PRINT_GENERAL,
; P 1568 7      P1 = FMT_DDP_DATO,P2 = 3,
; P 1569 7      P3 = SCRATCH[.M],P4 = xX'3400',
; L 1570 7      P5 = .TEMP4);! M10 M11
; xPRINT:      Test 21, Subtest 0, Error 2
; 1571 6      END;
; 1572 6      IF (TEMP4 = .(SCRATCH[.M] + 2)<0,16>) NEQ xX'0012' ! A10
; 1573 6      THEN ! A10
; 1574 7      BEGIN ! A10
; P 1575 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
; P 1576 7      PRLINK = PRINT_GENERAL,
; P 1577 7      P1 = FMT_DDP_DATO,P2 = 3,
; P 1578 7      P3 = SCRATCH[.M],P4 = xX'0012',
; L 1579 7      P5 = .TEMP4);! A10 M11
; xPRINT:      Test 21, Subtest 0, Error 3
; 1580 6      END; ! A10
; 1581 6
; 1582 5      END;
; 1583 5
; 1584 5      END
; 1585 4      WHILE $DS_INLOOP;
; 1586 4
; 1587 3      END;
; 1588 3
; 1589 2      END;
; 1590 2
; 1591 1      $DS_ENDTEST;

```

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00078 \$:
00000000 00000003 00088

.ADDRESS P.AAP, P.AAQ, P.AAR, TEST_021
.LONG 3, 0

;
;


```

                                .PSECT $PLIT$,NOWRT,NOEXE,2
44 44 20 74 65 73 66 66 4F 20 65 74 79 42 19 0015 000B0 P.AAP: .WORD 21
74 73 65 54 20 4F 54 41 44 20 50 000B2 P.AAQ: .ASCII <25>\Byte Offset DDP DATO Test\
                                00000000 000C1
                                00000000 000CC P.AAR: .LONG 0

                                .PSECT TEST_021,NOWRT,2
                                OFFC 00000
                                5B 00000000G 9F 9E 00002
                                5A 0000G CF 9E 00009
                                MOVAB a#DS$ERRHARD, R11
                                MOVAB SCRATCH, R10
;
; Call the Unibus sizer routine to locate any Unibus memory. T
; his builds
; a table of useable map entries. In the test which follows, t
; he map
; selected is drawn from this table.
;
0000G CF 0000G 00 FB 0000E CALLS #0, UNIBUS_SIZER ; 1484
53 0000G CF 3C 00013 MOVZWL FREE_MAP, MAP_NUMBER ; 1485
;
; Set up the UBI and UET for a DATO on a longword alignment and
; word
; alignment. The first DATO should write the data pattern into
; SCRATCH + 1,
; and the second DATO will write the data pattern into SCRATCH
; + 3. The
; latter DATO will wrap across a longword boundary, and thus ca
; use the UBI
; to perform two writes to memory.
;
57 D4 00018 CLRL M ; 1499
;
; Set up the UET address register with the Unibus address.
; Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs
; of the UBE map number. Bits 0 to 8 contain the byte number
; field of the CMI address.
;
50 0000G CF 0003F462 04 6A D4 0001A 1$: CLRL SCRATCH ; 1514
60 1234 8F C9 0001C CLRW SCRATCH+4 ; 1516
56 07 09 53 F0 0001F BISL3 #259170, UNIBUS_SPACE, R0
50 50 53 F0 00029 MOVW #4660, (R0) ; 1517
56 09 00 6A47 3E 0002E INSV MAP_NUMBER, #9, #7, UNIBUS_ADDR ; 1518
50 0000G CF 0003F460 50 F0 00033 MOVAW SCRATCH[M], R0 ; 1519
60 8F C9 00037 INSV R0, #0, #9, UNIBUS_ADDR
8F C9 0003C BISL3 #259168, UNIBUS_SPACE, R0
56 B0 00046 MOVW UNIBUS_ADDR, (R0) ; 1520
;
; Write the two MSBs of the Unibus address with the two MSBs of
; the map number.

```

```

55          53          02          07 EF 00049 ; EXTZV #7, #2, MAP_NUMBER, TEMP3 ; 1531
          55          08 C4 0004E ; MULL2 #8, TEMP3 ;
;
; Set up the UBI map. Put the CMI page frame number (PFN)
; into the register BUF_PFN. Then call the map setup routine
; with the parameters map number, PFN of the CMI buffer, byte
; offset mode (1 = byte offset mode enabled), and data path
; number (0 for DDP).
;
52          54          0F          6A47 3E 00051 ; MOVAV SCRATCH[M], TEMP ; 1545
          54          7E          09 EF 00055 ; EXTZV #9, #15, TEMP, BUF_PFN ; 1546
          0000G CF          01 7D 0005A ; MOVQ #1, -(SP) ; 1547
          52          53 DD 0005D ; PUSHL BUF_PFN ;
          53 DD 0005F ; PUSHL MAP_NUMBER ;
          04 FB 00061 ; CALLS #4, MAP_SETUP ;
;
; Execute the DATO and after a nominal wait check that the
; CMI destination contains the correct data. If not, report
; the error.
;
50          0000G CF 0003F464 8F C9 00066 ; BISL3 #259172, UNIBUS_SPACE, R0 ; 1557
60          55          05 A9 00070 ; BISW3 #5, TEMP3, (R0) ; 1559
          7E          0A 7D 00074 ; MOVQ #10, -(SP) ; 1561
          00000000G 9F          02 FB 00077 ; CALLS #2, @DS$WAITUS ;
50          0000G CF 0003F464 8F C9 0007E ; BISL3 #259172, UNIBUS_SPACE, R0 ; 1562
          0000' CF          60 B0 00088 ; MOVW (R0), STATUS0 ;
          0000' CF          8F AA 0008D ; BICW2 #65310, STATUS0 ;
          50          FF1E' CF          32 00094 ; CVTWL STATUS0, R0 ;
          1D 13 00099 ; BEQL 2$ ;
          7E 7C 0009B ; CLRQ -(SP) ;
          50 DD 0009D ; PUSHL R0 ;
          7E          02 7D 0009F ; MOVQ #2, -(SP) ;
          0000G CF 9F 000A2 ; PUSHAB FMT_UET_STAT ;
          0000G CF 9F 000A6 ; PUSHAB PRINT_GENERAL ;
          0000G CF 9F 000AA ; PUSHAB UBIMOD_DDP ;
          7E          0000G CF 9A 000AE ; MOVZBL LUN, -(SP) ;
          01 DD 000B3 ; PUSHL #1 ;
          6B          0A FB 000B5 ; CALLS #10, DS$ERRHARD ;
          58          6A47 3E 000B8 2$ : ; 1563
          50          68 3C 000BC ; MOVZWL (R8), R0 ;
          59          50 B0 000BF ; MOVW R0, TEMP4 ;
          3400 8F          50 B1 000C2 ; CMPW R0, #13312 ;
          24 13 000C7 ; BEQL 3$ ;
          7E D4 000C9 ; CLRL -(SP) ; 1570
          7E          59 3C 000CB ; MOVZWL TEMP4, -(SP) ;
          7E          3400 8F 3C 000CE ; MOVZWL #13312, -(SP) ;
          58 DD 000D3 ; PUSHL R8 ;
          03 DD 000D5 ; PUSHL #3 ;
          0000G CF 9F 000D7 ; PUSHAB FMT_DDP_DATO ;
          0000G CF 9F 000DB ; PUSHAB PRINT_GENERAL ;
          0000G CF 9F 000DF ; PUSHAB UBIMOD_DDP ;
          7E          0000G CF 9A 000E3 ; MOVZBL LUN, -(SP) ;
          02 DD 000E8 ; PUSHL #2 ;
          6B          0A FB 000EA ; CALLS #10, DS$ERRHARD ;

```

ZZ-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_021: Byte Offset DDP DATO Test
TEST_021: Byte Offset DDP DATO Test

K 13

6-Sep-1985

Fiche 2

Frame K13

Sequence 372

6-Sep-1985 17:20:57

VAX-11 Bliss-32 V4.1-746

Page 62

6-Sep-1985 17:15:01

[LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5

(8)

50	02	AA47	3C	000ED	3\$:	MOVZWL	SCRATCH+2[M], R0	:	1572
59		50	B0	000F2		MOVW	R0, TEMP4	:	
12		50	B1	000F5		CMPL	R0, #18	:	
		21	13	000F8		BEQL	4\$	:	
		7E	D4	000FA		CLRL	-(SP)	:	1579
7E		59	3C	000FC		MOVZWL	TEMP4, -(SP)	:	
		12	DD	000FF		PUSHL	#18	:	
		58	DD	00101		PUSHL	R8	:	
		03	DD	00103		PUSHL	#3	:	
	0000G	CF	9F	00105		PUSHAB	FMT_DDP_DATO	:	
	0000G	CF	9F	00109		PUSHAB	PRINT_GENERAL	:	
	0000G	CF	9F	0010D		PUSHAB	UBIMOD_DDP	:	
7E	0000G	CF	9A	00111		MOVZBL	LUN, -(SP)	:	
		03	DD	00116		PUSHL	#3	:	
6B		0A	FB	00118		CALLS	#10, DS\$ERRHARD	:	
00000000G	9F	00	FB	0011B	4\$:	CALLS	#0, a#DS\$INLOOP	:	1585
	03	50	E9	00122		BLBC	R0, 6\$	:	
		FEF2	31	00125	5\$:	BRW	1\$	:	
F9		01	F3	00128	6\$:	AOBLEQ	#1, M, 5\$	:	1499
00000000G	57	00	FB	0012C		CALLS	#0, a#DS\$BREAK	:	1589
	9F	01	D0	00133		MOVL	#1, R0	:	1591
	50		04	00136		RET		:	

; Routine Size: 311 bytes, Routine Base: TEST_021 + 0000


```
; 1592 2 $DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Byte Offset BDP DATO Test');
; 1593 2
; 1594 2
; 1595 2 !++
; 1596 2 ! TEST DESCRIPTION:
; 1597 2 !
; 1598 2 ! This test verifies the ability of the UBI to handle DATO transactions
; 1599 2 ! through each of the buffered data paths (BDPs) with byte offset mode
; 1600 2 ! enabled. The behavior of the UBI in this mode is similar to its
; 1601 2 ! behavior in a normal DATO, with the exception that odd byte addressing
; 1602 2 ! of the CMI is accomplished in this mode.
; 1603 2 !
; 1604 2 ! The test algorithm is as follows. Purge the BDP, clear two CMI
; 1605 2 ! locations, load the UET data register with 5678 (hex), and
; 1606 2 ! then execute a DATO with byte offset mode enabled. Change the data in
; 1607 2 ! the UET data register to 1234 (hex), and execute another DATO with
; 1608 2 ! Unibus address bit one changed from a 0 to a one. After the first
; 1609 2 ! DATO, both CMI longwords should still contain zero. After the second
; 1610 2 ! DATO, the first CMI location should contain 34567800 (hex), and the
; 1611 2 ! second CMI location should still be zero. At this time, the BDP is
; 1612 2 ! purge, and the two CMI locations are examined again. The first
; 1613 2 ! location should not have changed in value, but the second should now
; 1614 2 ! contain 00000012. This effectively tests the correct functionality of
; 1615 2 ! the byte flags.
; 1616 2 ! ASSUMPTIONS:
; 1617 2 !
; 1618 2 ! Test 1-Test 15
; 1619 2 !--
; 1620 2
; 1621 2 REGISTER
; 1622 2     BUF_PFN,
; 1623 2     M,
; 1624 2     MAP_NUMBER,
; 1625 2     TEMP,
; 1626 2     TEMP3,
; 1627 2     UBUS_ADDR: WORD;
; 1628 2
; 1629 2 MACRO
; 1630 2     UBUS_PF = UBUS_ADDR<9,7>x,          ! Defines the Unibus page frame field
; 1631 2     UBUS_BO = UBUS_ADDR<0,9>x;          ! Defines the Unibus byte number field
; 1632 2
; 1633 2 CODECOMMENT
; 1634 2     'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 1635 2     'a table of useable map entries. In the test which follows, the map',
; 1636 2     'selected is drawn from this table.',
; 1637 2     '
; 1638 2     '
; 1639 3     BEGIN
; 1640 3
; 1641 3     UNIBUS_SIZER();
; 1642 3
; 1643 2     END;
; 1644 2
; 1645 2 CODECOMMENT
; 1646 2     '
; 1646 2
```

```
; 1647 2 'Clear two successive longword aligned CMI locations. Put a unique data',
; 1648 2 'pattern into the UET data register. Set up the UBI for a DATO',
; 1649 2 'addressing the first CMI longword, execute it, and check that both CMI',
; 1650 2 'destinations still contain zero. Change the data in the UET data register',
; 1651 2 'and the Unibus address bit 1 from a zero to a one, and execute the second',
; 1652 2 'DATO. After a nominal wait, check that both CMI locations contain the',
; 1653 2 'expected data. Then execute a purge, and check that the upper longword',
; 1654 2 'now contains the remaining byte of the pattern.',
; 1655 2 ''
; 1656 3 BEGIN
; 1657 3
; 1658 3 INCR N FROM 1 TO 3 DO
; 1659 4 BEGIN
; 1660 4
; 1661 4 DO ! Scope loop begins here
; 1662 5 BEGIN
; 1663 5
; 1664 5 UBI_CSR(.N) = PURGE;
; 1665 5
; 1666 5 CODECOMMENT
; 1667 5 ''
; 1668 5 'Set up the UET address register with the Unibus',
; 1669 5 'address. Bits 9 to 15 (the Unibus page frame) contain',
; 1670 5 'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
; 1671 5 'the byte number field of the CMI address.',
; 1672 5 ''
; 1673 6 BEGIN
; 1674 6
; 1675 6 SCRATCH[0] = 0;
; 1676 6 SCRATCH[1] = 0;
; 1677 6 UET_DATA_REG = xX'5678';
; 1678 6 MAP_NUMBER = .FREE_MAP(.N);
; 1679 6 UBUS_PF = .MAP_NUMBER;
; 1680 6 UBUS_BO = SCRATCH[0];
; 1681 6 UET_ADDR_REG = .UBUS_ADDR;
; 1682 6
; 1683 5 END;
; 1684 5
; 1685 5 CODECOMMENT
; 1686 5 ''
; 1687 5 'Write the 2 MSBs of the Unibus address with the 2 MSBs of',
; 1688 5 'the map number.',
; 1689 5 ''
; 1690 6 BEGIN
; 1691 6
; 1692 6 TEMP3 = .MAP_NUMBER<7,2> * 8;
; 1693 6
; 1694 5 END;
; 1695 5
; 1696 5 CODECOMMENT
; 1697 5 ''
; 1698 5 'Set up the UBI map. Put the CMI page frame number (PFN)',
; 1699 5 'into the register BUF_PFN. Then call the map setup routine',
; 1700 5 'with the parameters map number, PFN of the CMI buffer',
; 1701 5 'address offset enabled, and data path number.'
```

```

; 1702 5
; 1703 6
; 1704 6
; 1705 6
; 1706 6
; 1707 6
; 1708 6
; 1709 5
; 1710 5
; 1711 5
; 1712 5
; 1713 5
; 1714 5
; 1715 5
; 1716 6
; 1717 6
; 1718 6
; 1719 6
; 1720 6
; L 1721 6
; xPRINT: Test 22, Subtest 0, Error 1
; L 1722 7
; xPRINT: Test 22, Subtest 0, Error 2
; 1723 6
; 1724 6
; 1725 7
; P 1726 7
; P 1727 7
; P 1728 7
; L 1729 7
; xPRINT: Test 22, Subtest 0, Error 3
; 1730 6
; 1731 6
; 1732 6
; 1733 7
; P 1734 7
; P 1735 7
; P 1736 7
; L 1737 7
; xPRINT: Test 22, Subtest 0, Error 4
; 1738 6
; 1739 6
; 1740 5
; 1741 5
; 1742 5
; 1743 5
; 1744 5
; 1745 5
; 1746 5
; 1747 6
; 1748 6
; 1749 6
; 1750 6
; 1751 6
; 1752 5

      ``:
      BEGIN
          TEMP = SCRATCH[0];
          BUF_PFN = .TEMP<9,15>;
          MAP_SETUP(.MAP_NUMBER,.BUF_PFN,1,.N);
      END;
CODECOMMENT
``:
'Execute the first DATO and after a nominal wait check that the',
'two CMI locations are still empty.',
``:
      BEGIN
          UET_CTRL_REG = .TEMP3 OR DATO;
          $DS_WAITUS(TIME=10);
          UBI_STATUS(.N,UBIMOD_BDP);          ! M7
          UET_STATUS(UBIMOD_BDP);          ! M7
          IF .SCRATCH[0] NEQ 0
          THEN
              BEGIN
                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                              PRLINK = PRINT_GENERAL,
                              P1 = FMT_BDP_DATO,P2 = 3,
                              P3 = .N,P4 = 0,P5 = .SCRATCH[0]);
                  Test 22, Subtest 0, Error 3
              END;
          IF .SCRATCH[1] NEQ 0
          THEN
              BEGIN
                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                              PRLINK = PRINT_GENERAL,
                              P1 = FMT_BDP_DATO,P2 = 3,
                              P3 = .N,P4 = 0,P5 = .SCRATCH[1]);
                  Test 22, Subtest 0, Error 4
              END;
          END;
CODECOMMENT
``:
'Change the data in the UET data register and change',
'Unibus address bit 1 from a zero to a one.',
``:
      BEGIN
          UET_DATA_REG = %X'1234';
          UET_ADDR_REG = .UBUS_ADDR + 2;
      END;

```



```

; 1753 5
; 1754 5
; 1755 5
; 1756 5
; 1757 5
; 1758 5
; 1759 6
; 1760 6
; 1761 6
; 1762 6
; 1763 6
; L 1764 6
; xPRINT: Test 22, Subtest 0, Error 5
; L 1765 7
; xPRINT: Test 22, Subtest 0, Error 6
; 1766 6
; 1767 6
; 1768 7
; P 1769 7
; P 1770 7
; P 1771 7
; L 1772 7
; xPRINT: Test 22, Subtest 0, Error 7
; 1773 6
; 1774 6
; 1775 6
; 1776 7
; P 1777 7
; P 1778 7
; P 1779 7
; L 1780 7
; xPRINT: Test 22, Subtest 0, Error 8
; 1781 6
; 1782 6
; 1783 5
; 1784 5
; 1785 5
; 1786 5
; 1787 5
; 1788 5
; 1789 5
; 1790 6
; 1791 6
; 1792 6
; 1793 6
; 1794 6
; 1795 6
; 1796 7
; P 1797 7
; P 1798 7
; P 1799 7
; L 1800 7
; xPRINT: Test 22, Subtest 0, Error 9
; 1801 6
; 1802 6

```

```

CODECOMMENT
'Execute the DATO and after a nominal wait check that the',
'two CMI locations contain the expected data.',
'
BEGIN
    UET_CTRL_REG = .TEMP3 OR DATO;

    $DS_WAITUS(TIME=10);
    UBI_STATUS(.N,UBIMOD_BDP);      ! M7
    UET_STATUS(UBIMOD_BDP);        ! M7
    IF .SCRATCH[0] NEQ %X'34567800'
    THEN
        BEGIN
            $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                PRLINK = PRINT_GENERAL,
                P1 = FMT_BDP_DATO,P2 = 3,
                P3 = .N,P4 = %X'34567800',P5 = .SCRATCH[0]);
        END;
    IF .SCRATCH[1] NEQ 0
    THEN
        BEGIN
            $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                PRLINK = PRINT_GENERAL,
                P1 = FMT_BDP_DATO,P2 = 3,
                P3 = .N,P4 = 0,P5 = .SCRATCH[1]);
        END;
    END;
CODECOMMENT
'Now purge the BDP and check that the upper CMI longword now',
'contains the remaining byte.',
'
BEGIN
    UBI_CSR(.N) = PURGE;
    $DS_WAITUS(TIME=10);
    IF .SCRATCH[1] NEQ %X'00000012'
    THEN
        BEGIN
            $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                PRLINK = PRINT_GENERAL,
                P1 = FMT_BDP_DATO,P2 = 3,
                P3 = .N,P4 = %X'00000012',P5 = .SCRATCH[1]);
        END;
    END;

```

ZZ-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_022: Byte Offset BDP DATO Test
TEST_022: Byte Offset BDP DATO Test

C 14

6-Sep-1985

Fiche 2

Frame C14

Sequence 377

6-Sep-1985 17:20:57

VAX-11 Bliss-32 V4.1-746

Page 67

6-Sep-1985 17:15:01

[LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5

(9)

```
; 1803 5      END;
; 1804 5
; 1805 5      END
; 1806 4      WHILE $DS_INLOOP;      ! Scope loop ends here
; 1807 4
; 1808 3      END;
; 1809 3
; 1810 2      END;
; 1811 2
; 1812 1      $DS_ENDTEST;
```

```
                                .PSECT DISPATCH,NOWRT,NOEXE,2
                                .ADDRESS P.AAS, P.AAT, P.AAU, TEST_022
00000000' 00000000' 00000000' 00000000' 00090 $: .LONG 3, 0
                                .PSECT $PLIT$,NOWRT,NOEXE,2
                                .WORD 22
44 42 20 74 65 73 66 66 4F 20 65 74 79 42 19 000D0 P.AAS: .ASCII <25>\Byte Offset BDP DATO Test\
74 73 65 54 20 4F 54 41 44 20 50 000D2 P.AAT: .LONG 0
                                00000000 000E1 P.AAU:
                                .PSECT TEST_022,NOWRT,2
                                .ENTRY TEST_022, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1592
                                OFFC 00000 R11
00000G CF 9E 00002 MOVAB STATUS0, R11
00000G CF 9E 00007 MOVAB SCRATCH, R10
59 00000000G 9F 9E 0000C MOVAB a#DS$ERRHARD, R9
; Call the Unibus sizer routine to locate any Unibus memory. T
; his builds
; a table of useable map entries. In the test which follows, t
; he map
; selected is drawn from this table.
                                0000G CF 00 FB 00013 CALLS #0, UNIBUS_SIZER ; 1641
; Clear two successive longword aligned CMI locations. Put a u
; nique data
; pattern into the UET data register. Set up the UBI for a DAT
; 0
; addressing the first CMI longword, execute it, and check that
; both CMI
; destinations still contain zero. Change the data in the UET
; data register
; and the Unibus address bit 1 from a zero to a one, and execut
; e the second
; DATO. After a nominal wait, check that both CMI locations co
; ntain the
```



```

;
; expected data. Then execute a purge, and check that the upper
; longword
; now contains the remaining byte of the pattern.
;
;
58      57      01  D0 00018      MOVL    #1, N                      ; 1658
50      57      02  78 0001B 1$:  ASHL    #2, N, R8                      ; 1664
50      58      CF  C1 0001F 2$:  ADDL3   UBI_UNDER_TEST, R8, R0
50      60      01  D0 00025      MOVL    #1, (R0)
;
; Set up the UET address register with the Unibus
; address. Bits 9 to 15 (the Unibus page frame) contain
; the 7 LSBs of the UBI map number. Bits 0 to 8 contain
; the byte number field of the CMI address.
;
50      0000G  CF  0003F462  6A  7C 00028      CLRQ    SCRATCH                      ; 1675
50      60      8F  C9 0002A      BISL3   #259170, UNIBUS_SPACE, R0      ; 1676
50      53      8F  B0 00034      MOVW    #22136, (R0)                      ; 1677
56      07      0000GCF47  3C 00039      MOVZWL  FREE_MAP[N], MAP_NUMBER      ; 1678
56      09      53  F0 0003F      INSV     MAP_NUMBER, #9, #7, UNIBUS_ADDR    ; 1679
50      50      6A  9E 00044      MOVAB    SCRATCH, R0                      ; 1680
50      0000G  CF  0003F460  50  F0 00047      INSV     R0, #0, #9, UNIBUS_ADDR    ;
50      60      8F  C9 0004C      BISL3   #259168, UNIBUS_SPACE, R0      ;
50      56      56  B0 00056      MOVW    UNIBUS_ADDR, (R0)                ; 1681
;
; Write the 2 MSBs of the Unibus address with the 2 MSBs of
; the map number.
;
55      53      02      07  EF 00059      EXTZV   #7, #2, MAP_NUMBER, TEMP3      ; 1692
55      55      08  C4 0005E      MULL2    #8, TEMP3
;
; Set up the UBI map. Put the CMI page frame number (PFN)
; into the register BUF_PFN. Then call the map setup routine
; with the parameters map number, PFN of the CMI buffer,
; address offset enabled, and data path number.
;
52      54      54      6A  9E 00061      MOVAB    SCRATCH, TEMP                      ; 1705
52      0F      09  EF 00064      EXTZV   #9, #15, TEMP, BUF_PFN              ; 1706
52      57      DD 00069      PUSHL     N                      ; 1707
52      01      DD 0006B      PUSHL     #1
52      52      DD 0006D      PUSHL     BUF_PFN
52      53      DD 0006F      PUSHL     MAP_NUMBER
52      0000G  CF  04  FB 00071      CALLS    #4, MAP_SETUP
;
; Execute the first DATO and after a nominal wait check that the
; two CMI locations are still empty.
;
50      0000G  CF  0003F464  8F  C9 00076      BISL3   #259172, UNIBUS_SPACE, R0      ; 1716
50      60      55  A9 00080      BISW3   #5, TEMP3, (R0)                ; 1718
50      7E      0A  7D 00084      MOVQ    #10, -(SP)                      ; 1720
04      AB      00000000G  9F      02  FB 00087      CALLS    #2, a#DS$WAITUS
04      0000GDF47  1FFFFFFE  8F  CB 0008E      BICL3   #536870910, aUBI_UNDER_TEST[N], STATUS1 ; 1721
50      50      AB  D0 0009A      MOVL     STATUS1, R0
50      1C      18 0009E      BGEQ    3$
50      7E      7C 000A0      CLRQ    -(SP)

```


[illegible]

```

50      0000G  CF 0003F460  8F C9 0014A      BISL3  #259168, UNIBUS_SPACE, R0      ;
60      56      02 A1 00154      ADDW3  #2, UBUS_ADDR, (R0)      ; 1750
;
; Execute the DATO and after a nominal wait check that the
; two CMI locations contain the expected data.
;
50      0000G  CF 0003F464  8F C9 00158      BISL3  #259172, UNIBUS_SPACE, R0      ; 1759
60      55      05 A9 00162      BISW3  #5, TEMP3, (R0)      ; 1761
7E      7E      0A 7D 00166      MOVQ   #10, -(SP)      ; 1763
04      00000000G  9F      02 FB 00169      CALLS  #2, @DS$WAITUS      ;
AB      0000GDF47 1FFFFFFE  8F CB 00170      BICL3  #536870910, @UBI_UNDER_TEST[N], STATUS1 ; 1764
50      04      AB      50      1C 18 00180      MOVL  STATUS1, R0      ;
7E      7E      7E 7C 00182      BGEQ   7$      ;
7E      D4 00184      CLRQ   -(SP)      ;
50      DD 00186      CLRL   -(SP)      ;
7E      D4 00188      PUSHL  R0      ;
57      DD 0018A      CLRL   -(SP)      ;
0000G  CF 9F 0018C      PUSHL  N      ;
0000G  CF 9F 00190      PUSHAB PRINT_CSR_ERR      ;
0000G  CF 9A 00194      PUSHAB UBIMOD_BDP      ;
7E      05 DD 00199      MOVZBL LUN, -(SP)      ;
69      0A FB 0019B      PUSHL  #5      ;
50      0000G  CF 0003F464  8F C9 0019E  7$:      ; 1765
6B      60 B0 001A8      CALLS  #10, DS$ERRHARD      ;
6B      FF1E  8F AA 001AB      BISL3  #259172, UNIBUS_SPACE, R0      ;
50      6B 32 001B0      MOVW   (R0), STATUS0      ;
1D      13 001B3      BICW2  #65310, STATUS0      ;
7E      7C 001B5      CVTWL  STATUS0, R0      ;
50      DD 001B7      BEQL   8$      ;
02      7D 001B9      CLRQ   -(SP)      ;
0000G  CF 9F 001BC      PUSHL  R0      ;
0000G  CF 9F 001C0      MOVQ   #2, -(SP)      ;
0000G  CF 9F 001C4      PUSHAB FMT_UET_STAT      ;
7E      06 DD 001CD      PUSHAB PRINT_GENERAL      ;
69      0A FB 001CF      PUSHAB UBIMOD_BDP      ;
34567800 8F 6A D1 001D2  8$:      ; 1766
24      13 001D9      MOVZBL LUN, -(SP)      ;
7E      D4 001DB      PUSHL  #6      ;
6A      DD 001DD      CALLS  #10, DS$ERRHARD      ;
34567800 8F DD 001DF      CMPL  SCRATCH, #878082048      ;
57      DD 001E5      BEQL   9$      ; 1772
03      DD 001E7      CLRL   -(SP)      ;
0000G  CF 9F 001E9      PUSHL  SCRATCH      ;
0000G  CF 9F 001ED      PUSHL  #878082048      ;
0000G  CF 9F 001F1      PUSHL  N      ;
7E      07 DD 001FA      PUSHL  #3      ;
69      0A FB 001FC      PUSHAB FMT_BDP_DATO      ;
50      04      AA D0 001FF  9$:      ; 1774
20      13 00203      PUSHAB PRINT_GENERAL      ;
7E      D4 00205      PUSHAB UBIMOD_BDP      ;
50      DD 00207      MOVZBL LUN, -(SP)      ;
7E      D4 00209      PUSHL  #7      ;
      BEQL  10$      ; 1780
      CLRL  -(SP)      ;
      PUSHL R0      ;
      CLRL  -(SP)      ;

```

[illegible]

```
; Routine Size: 641 bytes,    Routine Base: TEST_022 + 0000
```


ZZ-ECCBA-1.4
UBI_TESTS_16_22
01-04

TEST_022: Byte Offset BDP DATO Test
TEST_022: Byte Offset BDP DATO Test

H 14

6-Sep-1985

Fiche 2

Frame H14

Sequence 382

6-Sep-1985 17:20:57

VAX-11 Bliss-32 V4.1-746

Page 72

6-Sep-1985 17:15:01

[LEMIEUX.ECCBA.RELEASE]ECCBA6.B32;5

(10)

; 1813 1 \$DS_ENDMOD;
; 1814 1 END
; 1815 0 ELUDOM

.PSECT \$TSTCNT,NOWRT,NOEXE, OVR,2

00000016 00000 L_TSTCNT:

.LONG 22

PSECT SUMMARY

Name	Bytes	Attributes
\$OWN\$	22	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	240	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
DISPATCH	168	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_016	836	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_017	1183	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_018	595	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_019	536	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_020	357	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_021	311	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_022	641	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$TSTCNT	4	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, OVR, NOPIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
SYS\$COMMON:[SYSLIB]STARLET.L32;3	10093	1	0	598	00:00.8
SYS\$COMMON:[SYSLIB]DIAG.L32;265	784	17	2	85	00:00.4

COMMAND QUALIFIERS

; BLISS/LIST ECCBA6.B32

; Size: 4459 code + 434 data bytes
; Run Time: 00:45.4
; Elapsed Time: 03:39.3
; Lines/CPU Min: 2399

ZZ-ECCBA-1.4 TEST_022: Byte Offset BDP DATO Test
UBI_TESTS_16_22
01-04 TEST_022: Byte Offset BDP DATO Test

; Lexemes/CPU-Min: 31195
; Memory Used: 370 pages
; Compilation Complete

I 14

6-Sep-1985

Fiche 2 Frame I14

Sequence 383

6-Sep-1985 17:20:57

VAX-11 Bliss-32 V4.1-746

Page 73

```

: 0001 0  MODULE UBI_TESTS_23_26 (      ! UBI diagnostic program tests 23 to 26
: 0002 0      IDENT = '01-04'
: 0003 0      ) =
: 0004 1  BEGIN
: 0005 1
: 0006 1  !
: 0007 1  ! COPYRIGHT (C) 1980
: 0008 1  ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
: 0009 1  !
: 0010 1  ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
: 0011 1  ! COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
: 0012 1  ! ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
: 0013 1  ! MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
: 0014 1  ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
: 0015 1  ! TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
: 0016 1  ! REMAIN IN DEC.
: 0017 1  !
: 0018 1  ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
: 0019 1  ! AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
: 0020 1  ! CORPORATION.
: 0021 1  !
: 0022 1  ! DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
: 0023 1  ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
: 0024 1  !
: 0025 1  !
: 0026 1  ! **
: 0027 1  ! FACILITY:      VAX-11 Diagnostic Library
: 0028 1  !
: 0029 1  ! ABSTRACT:      This module contains tests 23 to 29 of the COMET
: 0030 1  !                  Unibus Interface Diagnostic Program.
: 0031 1  !
: 0032 1  ! ENVIRONMENT:    VAX-11 Diagnostic Supervisor
: 0033 1  !
: 0034 1  ! AUTHOR:         Rand Gray, CREATION DATE: 11-Dec-78
: 0035 1  !
: 0036 1  ! MODIFIED BY:    Don Monroe
: 0037 1  !
: 0038 1  !             0-7    May 1979
: 0039 1  !                  Changed the format of the UBI MAP registers.
: 0040 1  !
: 0041 1  !             0-8    July 1979
: 0042 1  !                  Modified to support the real UET module.
: 0043 1  !
: 0044 1  !                  Fixed a BUG in XFER_SETUP routine. Loop that setup the maps
: 0045 1  !                  would not work if MAP_NUM was not zero.
: 0046 1  !
: 0047 1  !             0-9    July 1979
: 0048 1  !                  Changed referances to 'Byte Offset' to 'Byte Number'
: 0049 1  !
: 0050 1  !
: 0051 1  !             0-10   October 1979
: 0052 1  !                  Added a return value to the MAP_BUFFERS routine to support
: 0053 1  !                  the changes to Module 7 Revision 10.
: 0054 1  !                  Added byte-offset parameter to MAP_BUFFERS routine so routine
: 0055 1  !                  returns with one additional page per buffer.

```



```
; 0056 1
; 0057 1
; 0058 1
; 0059 1
; 0060 1
; 0061 1
; 0062 1
; 0063 1
; 0064 1
; 0065 1
; 0066 1
; 0067 1
; 0068 1
; 0069 1
; 0070 1
; 0071 1
; 0072 1
; 0073 1
; 0074 1
; 0075 1
; 0076 1
; 0077 1
; 0078 1
; 0079 1
; 0080 1
; 0081 1
; 0082 1
; 0083 1
; 0084 1
; 0085 1
; 0086 1
; 0087 1

0-11 October 22,1979
      Added a routine to print MAP errors. Called via PRINT LINK
      in error hard calls.

0-12 December 7,1979
      Fixed bug in PROBE_ADDRESS routine in the ADAWI builtin.

0-13 April 7,1980
      Changed address of unibus space so second UBI will work.

0-14 April 8,1980
      Added a print general routine and converted PRINTX after
      error calls to print links.

1-00 June 16,1980
      Initial Release

1-01 July 31,1980
      Fixed bug in MAP_BUFFERS routine that would not detect
      insufficient memory for the requested number of buffers.

1-02 December 29,1980
      Fixed bug in MAP_BUFFERS routine. Supervisor allocates memory
      even if not enough for requested buffer size.

1-03 May 6, 1981
      Fixed bug in map_buffers routine.

Kevin LeMieux

1-04 February,1985
      Fixed bug in INIT code. If two DW's were specified; after the
      first pass one of them one of them was no longer tested.
      Fixed minor bugs in TEST 29.
```

```

0088 1  !
0089 1  ! TABLE OF CONTENTS:
; 0090 1  !
; 0091 1  !      Byte Offset DDP DATOB Test
; 0092 1  !      Byte Offset BDP DATOB Test
; 0093 1  !      Map Entry Functional Test
; 0094 1  !      CSR Status Bit Test
; 0095 1  !
; 0096 1  !
; 0097 1  !
; 0098 1  ! INCLUDE FILES:
; 0099 1  !
; 0100 1  !
; 0101 1  ! LIBRARY 'SYS$LIBRARY:STARLET';
; 0102 1  ! LIBRARY 'SYS$LIBRARY:DIAG';
; 0103 1  !
; 0104 1  !
; 0105 1  ! MACROS:
; 0106 1  !
; 0107 1  !
; 0108 1  !
; 0109 1  ! EQUATED SYMBOLS:
; 0110 1  !
; 0111 1  !
; 0112 1  ! LITERAL
; 0113 1  !     CHC$_ENINT = 4,
; 0114 1  !     CHC$_DSINT = 5;
; 0115 1  !
; 0116 1  ! LITERAL
; 0117 1  !     ALL_ONES = %X'FFFFFFFF',
; 0118 1  !     ALL_ZEROS = 0,
; 0119 1  !
; 0120 1  !     BR4 = %X'100',
; 0121 1  !     BR5 = %X'200',
; 0122 1  !     BR6 = %X'400',
; 0123 1  !     BR7 = %X'800',
; 0124 1  !
; 0125 1  !     CLEAR_MEM_ERR = %X'E0000000',
; 0126 1  !     CSR_MASK = %X'E0000001',
; 0127 1  !
; 0128 1  !     DATI = 1,
; 0129 1  !     DATIP = 3,
; 0130 1  !     DATO = 5,
; 0131 1  !     DATOB = 7,
; 0132 1  !
; 0133 1  !     DIAG_CHK_MODE = %X'C0000000',
; 0134 1  !     DISABLE_ECC = %X'20000000',
; 0135 1  !     ENABLE_ECC = %X'20000000',
; 0136 1  !     NORMAL = %X'E0000000',
; 0137 1  !     ERR = BIT31,
; 0138 1  !     MAP_MASK = %X'82607FFF',
; 0139 1  !     MEMORY_CONT = %X'F20000',
; 0140 1  !     NON_XIST_NEXUS = %X'F40000',
; 0141 1  !     NXM = BIT30,
; 0142 1  !     PAGE_MODE = %X'80000000',

```

! A8

! D3

! D8

! M7

! A8

```

: 0143 1 PURGE = BIT0,
: 0144 1 UCE = BIT29;
: 0145 1
: 0146 1 ! OWN STORAGE:
: 0147 1 !
: 0148 1 OWN
: 0149 1 BYTE_PATTERN: VECTOR[4,BYTE], ! Data patterns loaded at run time
: 0150 1 STATUS0: SIGNED WORD, ! Used by the UET status macro
: 0151 1 STATUS1, ! Used by the UBI status macro
: 0152 1 WORD_PATTERN: VECTOR[5,WORD]; ! Data patterns loaded at run time
: 0153 1
: 0154 1
: 0155 1 ! EXTERNAL REFERENCES:
: 0156 1 !
: 0157 1 EXTERNAL ROUTINE
: 0158 1 BUFFER_SETUP,
: 0159 1 DECODER,
: 0160 1 ENCODER,
: 0161 1 MAP_BUFFERS,
: 0162 1 MAP_SETUP,
: 0163 1 PRINT_GENERAL:NOVALUE,
: 0164 1 PROBE_ADDRESS,
: 0165 1 UNIBUS_SIZER,
: 0166 1 XFER_SETUP;
: 0167 1
: 0168 1 EXTERNAL
: 0169 1 CHAN_STAT,
: 0170 1 CODEWORDS: VECTOR[9,WORD],
: 0171 1 DECODED_WORDS: VECTOR[9],
: 0172 1 EVENT_FLAG: BITVECTOR[4],
: 0173 1 FMT_BDP_DATI,
: 0174 1 FMT_BDP_DATO,
: 0175 1 FMT_BDP_DATO,
: 0176 1 FMT_BYTE_OFF,
: 0177 1 FMT_CMI_UB,
: 0178 1 FMT_CPU_RW,
: 0179 1 FMT_DDP_DATI,
: 0180 1 FMT_DDP_DATO,
: 0181 1 FMT_DDP_DATO,
: 0182 1 FMT_DP_SEL,
: 0183 1 FMT_MAP_ADDR,
: 0184 1 ! FMT_MAP_BIT, ! D13
: 0185 1 FMT_MAP_CS,
: 0186 1 ! FMT_MAP_DB, ! D8
: 0187 1 FMT_MAP_FAIL,
: 0188 1 ! FMT_MAP_FAIL1, ! A11 D13
: 0189 1 FMT_MAP_INV,
: 0190 1 FMT_MCHK,
: 0191 1 FMT_NO_INT,
: 0192 1 FMT_NO_MCHK, ! A8
: 0193 1 FMT_NO_UB, ! A8
: 0194 1 FMT_UAI,
: 0195 1 FMT_UB_CMI,
: 0196 1 FMT_UB_TO, ! A8
: 0197 1 FMT_UB_DUMP,

```



```

; 0198 1 FMT_UBI_DUMP,
; 0199 1 FMT_UET_PB, ! A8
; 0200 1 FMT_UET_STAT,
; 0201 1 FMT_UNX_INT1,
; 0202 1 FMT_VECTOR,
; 0203 1 FREE_MAP: VECTOR[4,WORD],
; 0204 1 HANDLER,
; 0205 1 INT_VECTOR,
; 0206 1 INTERRUPT_EXP: BYTE,
; 0207 1 INTERRUPT_OCC: BYTE,
; 0208 1 LUN: BYTE,
; 0209 1 NOISY_WORDS: VECTOR[9,WORD],
; 0210 1 NUM_UBE,
; 0211 1 PRINT_CSR_ERR,
; 0212 1 PRINT_DCMF_ERR,
; 0213 1 PRINT_MAP_ERR, ! A13
; 0214 1 UBE_INT_OCC: VECTOR[4,BYTE], ! A8
; 0215 1 UBE_SERVICE: VECTOR[4], ! A8
; 0216 1 UBE_VECTOR: VECTOR[4,WORD], ! A8
; 0217 1 UBIMOD,
; 0218 1 UBIMOD_BDP,
; 0219 1 UBIMOD_BYTE_OFF,
; 0220 1 UBIMOD_CMI,
; 0221 1 UBIMOD_CPU_RW,
; 0222 1 UBIMOD_CSR,
; 0223 1 UBIMOD_DDP,
; 0224 1 UBIMOD_DP_SEL,
; 0225 1 UBIMOD_MAP,
; 0226 1 UBIMOD_MAP_CS,
; 0227 1 UBIMOD_MAP_ADDR,
; 0228 1 UBIMOD_MAP_CTRL,
; 0229 1 UBIMOD-UA1,
; 0230 1 UBIMOD_UB_CMI,
; 0231 1 UBI_VECTOR:WORD,
; 0232 1 UETMOD, ! A8
; 0233 1 SCRATCH: VECTOR[128],
; 0234 1 UBE_BASE_ADDR: VECTOR[4],
; 0235 1 UBE_BUF_SIZE,
; 0236 1 UBE_INIT_DATA: VECTOR[4,WORD], ! A10
; 0237 1 UBE_MODE: VECTOR[4,BYTE], ! A10
; 0238 1 UBE_MAP_NO: VECTOR[4,WORD], ! A10
; 0239 1 UBE_DP_NO: VECTOR[4,BYTE], ! A10
; 0240 1 ! UBE_ERROR, ! D8
; 0241 1 UBE_EXP_DATA: VECTOR[4,WORD],
; 0242 1 UBE_XFER_ERR: BITVECTOR[4],
; 0243 1 UBE_STAT_ERR: BITVECTOR[4],
; 0244 1 UBE0_ISR,
; 0245 1 UBI_UNDER_TEST,
; 0246 1 UNIBUS_SPACE,
; 0247 1 UET_ISR,
; 0248 1 VALID_MAPS: BITVECTOR[512];
; 0249 1
; 0250 1 MACRO
; 0251 1 !++
; 0252 1 ! This macro defines the address of the control and status register

```

```

; 0253 1      ! for the given buffered data path number.
; 0254 1      !--
; 0255 1      UBI_CSR(BDP) = (.UBI_UNDER_TEST + BDP*4)x,
; 0256 1
; 0257 1      !++
; 0258 1      ! This macro defines the address of a UBI map for the given map
; 0259 1      ! number.
; 0260 1      !--
; 0261 1      UBI_MAP(NUM) = (.UBI_UNDER_TEST + xX'800' + NUM*4)x;
; 0262 1
; 0263 1      !++
; 0264 1      ! These macros temporarily define the addresses of the (simulated) UET
; 0265 1      ! registers. They will be replaced for the real UET.
; 0266 1      !--
; 0267 1      MACRO                                     ! A8
; 0268 1      UET_ADDR_REG = (.UNIBUS_SPACE OR xX'3F460')<0,16>x, ! A8 M15
; 0269 1      UET_DATA_REG = (.UNIBUS_SPACE OR xX'3F462')<0,16>x, ! A8 M15
; 0270 1      UET_DATA_REGB = (.UNIBUS_SPACE OR xX'3F462')<0,8>x, ! A8 M15
; 0271 1      UET_CTRL_REG = (.UNIBUS_SPACE OR xX'3F464')<0,16>x, ! A8 M15
; 0272 1      UET_CTRL_REGB = (.UNIBUS_SPACE OR xX'3F465')<0,8>x, ! A8 M15
; 0273 1
; 0274 1      !++
; 0275 1      ! This macro is used for checking the status of the UET.           ! A8
; 0276 1      !--
; M 0277 1      UET_STATUS(CALLOUT) =                                     ! A8
; M 0278 1      STATUS0 = .UET_CTRL_REG; ! Read the UET control register ! A8
; M 0279 1      STATUS0 = .STATUS0 AND xX'E1';                             ! A8
; M 0280 1      IF .STATUS0 NEQ 0                                           ! A8
; M 0281 1      THEN                                                         ! A8
; M 0282 1      BEGIN                                                         ! A8
; M 0283 1      $DS_ERRHARD(UNIT=.LUN,MSGADR=CALLOUT,
; M 0284 1      PRLINK = PRINT_GENERAL,
; M 0285 1      P1 = FMT_UET_STAT,P2 = 2,
; M 0286 1      P3 = 0,P4 = .STATUS0); ! A8
; 0287 1      END;x;                                                         ! A8
; 0288 1      MACRO                                                         ! A8
; 0289 1      !++
; 0290 1      ! This macro is used for checking the status of the UBI.
; 0291 1      !--
; M 0292 1      UBI_STATUS(BDP,CALLOUT) =
; M 0293 1      STATUS1 = .UBI_CSR(BDP) AND CSR_MASK;
; M 0294 1      IF .STATUS1 LSS 0
; M 0295 1      THEN
; M 0296 1      $DS_ERRHARD(UNIT=.LUN,
; M 0297 1      MSGADR=CALLOUT,
; M 0298 1      PRLINK=PRINT_CSR_ERR,
; 0299 1      P1=BDP,P2=0,P3=.STATUS1);x;
; 0300 1
; 0301 1      !
; 0302 1      ! Required Diagnostic Supervisor macros.
; 0303 1      !
; L 0304 1      $DS_BGNMOD(ENV=0,TEST=23);
; xPRINT: 1      Bliss-32 Diagnostic Macro Library version 7.0 "DIAG.L32 (57)"
; 0305 1      $DS_DSADEF;
; 0306 1      $DS_SECDEF(ALL,UBE);

```

ZZ-ECCBA-1.4 01-04
UBI TESTS_23_26
01-04

C 15

6-Sep-1985

Fiche 2

Frame C15

Sequence 390

6-Sep-1985 17:24:39

VAX-11 Bliss-32 V4.1-746

Page 7

6-Sep-1985 17:15:04

[LEMIEUX.ECCBA.RELEASE]ECCBA7.B32;5

(2)

; 0307 1
; 0308 1 BUILTIN
; 0309 1 ROT;

! This is so we can do ROTLs

TEST_023: Byte Offset DDP DATOB Test

```
; 0310 2 $DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Byte Offset DDP DATOB Test');
; 0311 2
; 0312 2 !**
; 0313 2 ! TEST DESCRIPTION:
; 0314 2 !
; 0315 2 !     This test verifies the ability of the UBI to handle DATOB transactions
; 0316 2 !     through the direct data path with byte offset mode enabled.
; 0317 2 !     This test is similar to the DDP DATOB test, except that in this test
; 0318 2 !     the ability to address odd CMI locations is verified.
; 0319 2 !
; 0320 2 ! ASSUMPTIONS:
; 0321 2 !
; 0322 2 !     Test 1-Test 15
; 0323 2 !--
; 0324 2
; 0325 2 LOCAL
; 0326 2     BUF_PFN,
; 0327 2     MAP_NUMBER,
; 0328 2     TEMP,
; 0329 2     TEMP3,
; 0330 2     TEMP4: WORD,
; 0331 2     TEMP5: WORD,
; 0332 2     UBUS_ADDR: WORD;
; 0333 2
; 0334 2 STACKLOCAL
; 0335 2     TEMP0:WORD;
; 0336 2
; 0337 2 MACRO
; 0338 2     UBUS_PF = UBUS_ADDR<9,7>%x,      ! Defines the Unibus page frame field
; 0339 2     UBUS_BO = UBUS_ADDR<0,9>%x;      ! Defines the Unibus byte number field
; 0340 2
; 0341 2 MAP
; 0342 2     SCRATCH: VECTOR[6,BYTE];
; 0343 2
; 0344 2 CODECOMMENT
; 0345 2 ''
; 0346 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 0347 2 'a table of useable map entries. In the test which follows, the map',
; 0348 2 'selected is drawn from this table.',
; 0349 2 '';
; 0350 3     BEGIN
; 0351 3
; 0352 3     UNIBUS_SIZER();
; 0353 3     MAP_NUMBER = .FREE_MAP[0];
; 0354 3
; 0355 2     END;
; 0356 2
; 0357 2 CODECOMMENT
; 0358 2 ''
; 0359 2 'Set up the UBI and UET for a DATOB on a longword alignment and byte',
; 0360 2 'alignment. The first DATOB should write the data pattern into SCRATCH + 1.',
; 0361 2 'and the second DATOB should write the data pattern into SCRATCH + 2.',
; 0362 2 '';
; 0363 3     BEGIN
; 0364 3
```

```

: 0365 3      INCR M FROM 0 TO 1 DO
: 0366 4      BEGIN
: 0367 4
: 0368 4      DO
: 0369 5      BEGIN
: 0370 5
: 0371 5      CODECOMMENT
: 0372 5      ''
: 0373 5      'Set up the UET address register with the Unibus address.',
: 0374 5      'Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs',
: 0375 5      'of the UBE map number. Bits 0 to 8 contain the byte number',
: 0376 5      'field of the CMI address.',
: 0377 5      ''
: 0378 6      BEGIN
: 0379 6
: 0380 6      SCRATCH[0] = -1;      ! Clear the memory destinations M12
: 0381 6      SCRATCH[1] = -1;      ! M12
: 0382 6      SCRATCH[2] = -1;      ! M12
: 0383 6      SCRATCH[3] = -1;      ! M12
: 0384 6      TEMP = XX'AB';
: 0385 6      TEMPO + .M = .TEMP;
: 0386 6      UET_DATA_REG = .TEMPO;
: 0387 6      UBUS_PF = .MAP_NUMBER;
: 0388 6      UBUS_BO = SCRATCH[.M];
: 0389 6      UET_ADDR_REG = .UBUS_ADDR;
: 0390 6
: 0391 5      END;
: 0392 5
: 0393 5      CODECOMMENT
: 0394 5      ''
: 0395 5      'Write the two MSBs of the Unibus address with the two MSBs of',
: 0396 5      'the map number.',
: 0397 5      ''
: 0398 6      BEGIN
: 0399 6
: 0400 6      TEMP3 = .MAP_NUMBER<7,2> * 8;
: 0401 6
: 0402 5      END;
: 0403 5
: 0404 5      CODECOMMENT
: 0405 5      ''
: 0406 5      'Set up the UBI map. Put the CMI page frame number (PFN)',
: 0407 5      'into the register BUF_PFN. Then call the map setup routine',
: 0408 5      'with the parameters map number, PFN of the CMI buffer, byte',
: 0409 5      'offset mode (1 = byte offset mode enabled), and data path',
: 0410 5      'number (0 for DDP).',
: 0411 5      ''
: 0412 6      BEGIN
: 0413 6
: 0414 6      TEMP = SCRATCH[.M];      ! Get the address of SCRATCH
: 0415 6      BUF_PFN = .TEMP<9,15>;
: 0416 6      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,1,0);
: 0417 6
: 0418 5      END;
: 0419 5

```

```

: 0420 5      CODECOMMENT
: 0421 5      ''
: 0422 5      'Execute the DATOB and after a nominal wait check that the',
: 0423 5      'CMI destination contains the correct data. If not, report',
: 0424 5      'the error.',
: 0425 5      ''
: 0426 6      BEGIN
: 0427 6
: 0428 6          UET_CTRL_REG = .TEMP3 OR DATOB;
: 0429 6
: 0430 6          $DS_WAITUS(TIME=10);
: L 0431 7      UET_STATUS(UBIMOD_DDP);          ! M7
: %PRINT:      Test 23, Subtest 0, Error 1
: 0432 6
: 0433 6          IF .M EQL 0                      ! A12
: 0434 6          THEN                          ! A12
: 0435 7              BEGIN                      ! A12
: 0436 7                  TEMP4 = %X'ABFF';      ! A12
: 0437 7                  TEMP5 = %X'FFFF';      ! A12
: 0438 7              END                      ! A12
: 0439 6          ELSE                          ! A12
: 0440 7              BEGIN                      ! A12
: 0441 7                  TEMP4 = %X'FFFF';      ! A12
: 0442 7                  TEMP5 = %X'FFAB';      ! A12
: 0443 6              END;                      ! A12
: 0444 6
: 0445 6          IF .(SCRATCH[0])<0,16> NEQ .TEMP4      ! M12
: 0446 6          THEN
: 0447 7              BEGIN
: P 0448 7                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
: P 0449 7                      PRLINK = PRINT_GENERAL,
: P 0450 7                      P1 = FMT_DDP_DATOB,P2 = 4,
: P 0451 7                      P3 = SCRATCH[0],P4 = .M,
: L 0452 7                      P5 = .TEMP4,P6 = .(SCRATCH[0])<0,16>); ! M12
: %PRINT:      Test 23, Subtest 0, Error 2
: 0453 6          END;
: 0454 6
: 0455 6          IF .(SCRATCH[2])<0,16> NEQ .TEMP5      ! M12
: 0456 6          THEN
: 0457 7              BEGIN
: P 0458 7                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
: P 0459 7                      PRLINK = PRINT_GENERAL,
: P 0460 7                      P1 = FMT_DDP_DATOB,P2 = 4,
: P 0461 7                      P3 = SCRATCH[2],P4 = .M,
: L 0462 7                      P5 = .TEMP5,P6 = .(SCRATCH[2])<0,16>); ! M12
: %PRINT:      Test 23, Subtest 0, Error 3
: 0463 6          END;
: 0464 6
: 0465 5          END;
: 0466 5
: 0467 5          END
: 0468 4          WHILE $DS_INLOOP;
: 0469 4
: 0470 3          END;
: 0471 3

```


ZZ-ECCBA-1.4 TEST_023: Byte Offset DDP DATOB Test
UBI_TESTS_23_26
01-04 TEST_023: Byte Offset DDP DATOB Test

G 15

6-Sep-1985 Fiche 2 Frame G15 Sequence 394
6-Sep-1985 17:24:39 VAX-11 Bliss-32 V4.1-746 Page 11
6-Sep-1985 17:15:04 [LEMIEUX.ECCBA.RELEASE]ECCBA7.B32;5 (3)

; 0472 2
; 0473 2
; 0474 1
END;
\$DS_ENDTEST;

.TITLE UBI_TESTS_23_26
.IDENT \01-04\

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00000 \$:
00000000 00000003 00010

.ADDRESS P.AAA, P.AAB, P.AAC, TEST_023
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

44 44 20 74 65 73 66 66 4F 20 65 74 79 42 1A 0017 00000 P.AAA: .WORD 23
74 73 65 54 20 42 4F 54 41 44 20 50 00002 P.AAB: .ASCII <26>\Byte Offset DDP DATOB Test\
00011
0001D .BLKB 3
00000000 00020 P.AAC: .LONG 0

.PSECT \$OWN\$,NOEXE,2

00000 BYTE_PATTERN:
.BLKB 4
00004 STATUS0: .BLKB 2
00006 .BLKB 2
00008 STATUS1: .BLKB 4
0000C WORD_PATTERN:
.BLKB 10

DSA\$AL_APTMAIL= 65024
DSA\$GL_FLAGS= 65024
DSA\$GL_APTCOM= 65028
DSA\$GL_PASSES= 65032
DSA\$GL_UNITS= 65036
DSA\$GL_SECTNO= 65040
DSA\$GL_SID= 65044
DSA\$GL_MSGTYP= 65088
DSA\$GL_ERRNO= 65092
DSA\$GL_EVENT= 65096
DSA\$GL_SUBTNO= 65100
DSA\$GL_TESTNO= 65104
DSA\$GL_PASSNO= 65108
DSA\$GL_DEVLEN= 65112
DSA\$GL_DEVNAM= 65116
DSA\$GL_MSGPTR= 65128
.EXTRN BUFFER_SETUP, DECODER
.EXTRN ENCODER, MAP_BUFFERS
.EXTRN MAP_SETUP, PRINT_GENERAL
.EXTRN PROBE_ADDRESS, UNIBUS_SIZER
.EXTRN XFER_SETUP, CHAN_STAT
.EXTRN CODEWORDS, DECODED_WORDS
.EXTRN EVENT_FLAG, FMT_BDP_DATI
.EXTRN FMT_BDP_DATO, FMT_BDP_DATOB

ZZ-ECCBA-1.4
UBI_TESTS_23_26
01-04

TEST_023: Byte Offset DDP DATOB Test
TEST_023: Byte Offset DDP DATOB Test

H 15

6-Sep

6-Sep

6-Sep

Fiche 2 Frame H15

Sequence 395

17:24:39

VAX-11 Bliss-32 V4.1-746

Page 12

17:15:04

[LEMIEUX.ECCBA.RELEASE]ECCBA7.B32;5

(3)

```
.EXTRN FMT_BYTE_OFF, FMT_CMI_UB
.EXTRN FMT_CPU_RW, FMT_DDP_DATI
.EXTRN FMT_DDP_DATO, FMT_DDP_DATOB
.EXTRN FMT_DP_SEL, FMT_MAP_ADDR
.EXTRN FMT_MAP_CS, FMT_MAP_FAIL
.EXTRN FMT_MAP_INV, FMT_MCHK
.EXTRN FMT_NO_INT, FMT_NO_MCHK
.EXTRN FMT_NO_UB, FMT_UAI
.EXTRN FMT_UB_CMI, FMT_UB_TO
.EXTRN FMT_UB_DUMP, FMT_UBI_DUMP
.EXTRN FMT_UET_PB, FMT_UET_STAT
.EXTRN FMT_UNX_INT1, FMT_VECTOR
.EXTRN FREE_MAP, HANDLER
.EXTRN INT_VECTOR, INTERRUPT_EXP
.EXTRN INTERRUPT_OCC, LUN
.EXTRN NOISY_WORDS, NUM_UB
.EXTRN PRINT_CSR_ERR, PRINT_DCOMP_ERR
.EXTRN PRINT_MAP_ERR, UBE_INT_OCC
.EXTRN UBE_SERVICE, UBE_VECTOR
.EXTRN UBIMOD, UBIMOD_BDP
.EXTRN UBIMOD_BYTE_OFF
.EXTRN UBIMOD_CMI, UBIMOD_CPU_RW
.EXTRN UBIMOD_CSR, UBIMOD_DDP
.EXTRN UBIMOD_DP_SEL, UBIMOD_MAP
.EXTRN UBIMOD_MAP_CS, UBIMOD_MAP_ADDR
.EXTRN UBIMOD_MAP_CTRL
.EXTRN UBIMOD_UAI, UBIMOD_UB_CMI
.EXTRN UBI_VECTOR, UETMOD
.EXTRN SCRATCH, UBE_BASE_ADDR
.EXTRN UBE_BUF_SIZE, UBE_INIT_DATA
.EXTRN UBE_MODE, UBE_MAP_NO
.EXTRN UBE_DP_NO, UBE_EXP_DATA
.EXTRN UBE_XFER_ERR, UBE_STAT_ERR
.EXTRN UBE0_ISR, UBI_UNDER_TEST
.EXTRN UNIBUS_SPACE, UET_ISR
.EXTRN VALID_MAPS, DS$WAITUS
.EXTRN DS$ERRHARD, DS$INLOOP
.EXTRN DS$BREAK
```

.PSECT TEST_023,NOWRT,2

.ENTRY TEST_023, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0310
R11

MOVAB @DS\$ERRHARD, R11

MOVAB SCRATCH, R10

SUBL2 #4, SP

; Call the Unibus sizer routine to locate any Unibus memory. T
his builds
; a table of useable map entries. In the test which follows, t
he map
; selected is drawn from this table.

CALLS #0, UNIBUS_SIZER
MOVZWL FREE_MAP, MAP_NUMBER

; 0352

; 0353

OFFC 00000

```
5B 00000000G 9F 9E 00002
5A 0000G CF 9E 00009
5E 04 C2 0000E
```

```
0000G CF 00 00 00011
56 0000G CF 3C 00016
```

```

;
;
;
;
; Set up the UBI and UET for a DATOB on a longword alignment and
; byte alignment. The first DATOB should write the data pattern into
; SCRATCH + 1,
; and the second DATOB should write the data pattern into SCRATCH
; + 2.
;
;          52 D4 0001B          CLRL M          ; 0365
;
; Set up the UET address register with the Unibus address.
; Bits 9 to 15 (the Unibus page frame) contain the 7 LSBs
; of the UBE map number. Bits 0 to 8 contain the byte number
; field of the CMI address.
;
;          6A          01 CE 0001D 1$: MNEGL #1, SCRATCH          ; 0380
;          55          AB 8F 9A 00020 MOVZBL #171, TEMP          ; 0384
;          02 AE42 9F 00024 PUSHAB TEMP0[M]          ; 0385
;          9E          55 D0 00028 MOVL TEMP, a(SP)+          ;
;          50 0000G CF 0003F462 8F C9 0002B BISL3 #259170, UNIBUS_SPACE, R0          ;
;          60          02 AE B0 00035 MOVW TEMP0, (R0)          ; 0386
;          07          09 56 F0 00039 INSV MAP_NUMBER, #9, #7, UBUS_ADDR          ; 0387
;          57          52 5A C1 0003E ADDL3 R10, M, R0          ; 0388
;          09          50 F0 00042 INSV R0, #0, #9, UBUS_ADDR          ;
;          57          00 50 F0 00042 BISL3 #259168, UNIBUS_SPACE, R0          ;
;          00          CF 0003F460 8F C9 00047 MOVW UBUS_ADDR, (R0)          ; 0389
;          50          60 57 B0 00051
;
; Write the two MSBs of the Unibus address with the two MSBs of
; the map number.
;
;          58          56          02          07 EF 00054          EXTZV #7, #2, MAP_NUMBER, TEMP3          ; 0400
;          58          08 C4 00059          MULL2 #8, TEMP3          ;
;
; Set up the UBI map. Put the CMI page frame number (PFN)
; into the register BUF_PFN. Then call the map setup routine
; with the parameters map number, PFN of the CMI buffer, byte
; offset mode (1 = byte offset mode enabled), and data path
; number (0 for DDP).
;
;          59          55          52          5A C1 0005C          ADDL3 R10, M, TEMP          ; 0414
;          55          0F 09 EF 00060          EXTZV #9, #15, TEMP, BUF_PFN          ; 0415
;          7E          01 7D 00065          MOVQ #1, -(SP)          ; 0416
;          0000G CF          0240 8F BB 00068          PUSHR #^M<R6,R9>          ;
;          04 FB 0006C          CALLS #4, MAP_SETUP          ;
;
; Execute the DATOB and after a nominal wait check that the
; CMI destination contains the correct data. If not, report
; the error.
;
;          50          0000G CF 0003F464 8F C9 00071          BISL3 #259172, UNIBUS_SPACE, R0          ; 0426
;          60          58 07 A9 0007B          BISW3 #7, TEMP3, (R0)          ; 0428
;          7E          0A 7D 0007F          MOVQ #10, -(SP)          ; 0430
;          00000000G 9F 02 FB 00082          CALLS #2, a#DS$WAITUS          ;
;          50          0000G CF 0003F464 8F C9 00089          BISL3 #259172, UNIBUS_SPACE, R0          ; 0431
;          0000' CF 60 B0 00093          MOVW (R0), STATUS0          ;

```


ZZ-ECCBA-1.4
UBI_TESTS_23_26
01-04

TEST_023: Byte Offset DDP DATOB Test
TEST_023: Byte Offset DDP DATOB Test

J 15

6-Sep-1985

Fiche 2 Frame J15

Sequence 397

6-Sep-1985 17:24:39

VAX-11 Bliss-32 V4.1-746

Page 14

6-Sep-1985 17:15:04

[LEMIEUX.ECCBA.RELEASE]ECCBA7.B32;5

(3)

0000'	CF	FF1E	8F	AA	00098	BICH2	#65310, STATUS0	:	
50		0000'	CF	32	0009F	CVTWL	STATUS0, R0	:	
			1D	13	000A4	BEQL	2\$	:	
			7E	7C	000A6	CLRQ	-(SP)	:	
			50	DD	000A8	PUSHL	R0	:	
7E			02	7D	000AA	MOVQ	#2, -(SP)	:	
	0000G		CF	9F	000AD	PUSHAB	FMT UET STAT	:	
	0000G		CF	9F	000B1	PUSHAB	PRINT GENERAL	:	
	0000G		CF	9F	000B5	PUSHAB	UBIMOD DDP	:	
7E	0000G		CF	9A	000B9	MOVZBL	LUN, -(SP)	:	
			01	DD	000BE	PUSHL	#1	:	
6B			0A	FB	000C0	CALLS	#10, DS\$ERRHARD	:	
			52	D5	000C3	TSTL	M	:	0433
			0A	12	000C5	BNEQ	3\$	:	
54	ABFF		8F	B0	000C7	MOVW	#-21505, TEMP4	:	0436
53			01	AE	000CC	MNEGW	#1, TEMP5	:	0437
			07	11	000CF	BRB	4\$	:	0433
54			01	AE	000D1	MNEGW	#1, TEMP4	:	0441
53	AB		8F	99	000D4	CVTBW	#-85, TEMP5	:	0442
54			6A	B1	000D8	CMPL	SCRATCH, TEMP4	:	0445
			22	13	000DB	BEQL	5\$	:	
7E			6A	3C	000DD	MOVZWL	SCRATCH, -(SP)	:	0452
7E			54	3C	000E0	MOVZWL	TEMP4, -(SP)	:	
			52	DD	000E3	PUSHL	M	:	
			5A	DD	000E5	PUSHL	R10	:	
			04	DD	000E7	PUSHL	#4	:	
	0000G		CF	9F	000E9	PUSHAB	FMT DDP DATOB	:	
	0000G		CF	9F	000ED	PUSHAB	PRINT GENERAL	:	
	0000G		CF	9F	000F1	PUSHAB	UBIMOD DDP	:	
7E	0000G		CF	9A	000F5	MOVZBL	LUN, -(SP)	:	
			02	DD	000FA	PUSHL	#2	:	
6B			0A	FB	000FC	CALLS	#10, DS\$ERRHARD	:	
53	02		AA	B1	000FF	CMPL	SCRATCH+2, TEMP5	:	0455
			24	13	00103	BEQL	6\$	:	
7E	02		AA	3C	00105	MOVZWL	SCRATCH+2, -(SP)	:	0462
7E			53	3C	00109	MOVZWL	TEMP5, -(SP)	:	
			52	DD	0010C	PUSHL	M	:	
	02		AA	9F	0010E	PUSHAB	SCRATCH+2	:	
			04	DD	00111	PUSHL	#4	:	
	0000G		CF	9F	00113	PUSHAB	FMT DDP DATOB	:	
	0000G		CF	9F	00117	PUSHAB	PRINT GENERAL	:	
	0000G		CF	9F	0011B	PUSHAB	UBIMOD DDP	:	
7E	0000G		CF	9A	0011F	MOVZBL	LUN, -(SP)	:	
			03	DD	00124	PUSHL	#3	:	
6B			0A	FB	00126	CALLS	#10, DS\$ERRHARD	:	
00000000G	9F		00	FB	00129	CALLS	#0, a#DS\$INLOOP	:	0468
	03		50	E9	00130	BLBC	R0, 8\$	:	
		FEE7	31	00133	BRW	1\$		:	
F9	52		01	F3	00136	AOBLEQ	#1, M, 7\$	:	0365
00000000G	9F		00	FB	0013A	CALLS	#0, a#DS\$BREAK	:	0472
	50		01	D0	00141	MOVL	#1, R0	:	0474
			04	00144	RET			:	

; Routine Size: 325 bytes, Routine Base: TEST_023 + 0000

ZZ-ECCBA-1.4 TEST_023: Byte Offset DDP DATOB Test
UBI TESTS_23_26
01-04 TEST_023: Byte Offset DDP DATOB Test

K 15

6-Sep-1985

Fiche 2

Frame K15

Sequence 398

6-Sep-1985 17:24:39

VAX-11 Bliss-32 V4.1-746

Page 15

6-Sep-1985 17:15:04

[LEMIEUX.ECCBA.RELEASE]ECCBA7.B32;5

(3)

```
; 0475 2 $DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Byte Offset BDP DATOB Test');
; 0476 2
; 0477 2
; 0478 2 !**
; 0479 2 ! TEST DESCRIPTION:
; 0480 2 !
; 0481 2 ! This test verifies the ability of the UBI to handle DATOB transactions
; 0482 2 ! through each of the buffered data paths (BDPs) with byte offset mode
; 0483 2 ! enabled. This test is similar to the BDP DATOB test, except that in
; 0484 2 ! this test the ability to address odd CMI locations is verified.
; 0485 2 !
; 0486 2 ! The test algorithm is as follows. Purge the BDP, clear two CMI
; 0487 2 ! locations, load the UET data register with FF78, and execute a DATOB
; 0488 2 ! with Unibus address bits A1 and A0 = 00. Check that the two CMI
; 0489 2 ! locations are still empty. Change the data to 56FF, and execute
; 0490 2 ! another DATOB, this time with Unibus A1 and A0 = 01. Check that the
; 0491 2 ! two CMI locations are still empty. Change the data to FF34, and
; 0492 2 ! execute another DATOB, this time with Unibus A1 and A0 = 10. Check
; 0493 2 ! that the CMI locations are still empty. Change the data to 12FF, and
; 0494 2 ! execute another DATOB, this time with Unibus A1 and A0 = 11. Check
; 0495 2 ! that the first CMI location now contains 34567800 (hex), and that the
; 0496 2 ! second CMI location is still empty. Now purge the BDP, and check that
; 0497 2 ! the data in the first CMI location is unchanged, but the data in the
; 0498 2 ! second CMI location is now 00000012. This tests the correct
; 0499 2 ! functionality of the byte flags.
; 0500 2 !
; 0501 2 ! ASSUMPTIONS:
; 0502 2 !
; 0503 2 ! Test 1-Test 15
; 0504 2 !
; 0505 2 !
; 0506 2 ! REGISTER
; 0507 2 ! BUF_PFN,
; 0508 2 ! MAP_NUMBER,
; 0509 2 ! TEMP,
; 0510 2 ! TEMP3,
; 0511 2 ! UBUS_ADDR: WORD;
; 0512 2 !
; 0513 2 ! MACRO
; 0514 2 ! UBUS_PF = UBUS_ADDR<9,7>x, ! Defines the Unibus page frame field
; 0515 2 ! UBUS_BO = UBUS_ADDR<0,9>x; ! Defines the Unibus byte number field
; 0516 2 !
; 0517 2 ! CODECOMMENT
; 0518 2 !
; 0519 2 ! 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 0520 2 ! 'a table of useable map entries. In the test which follows, the map',
; 0521 2 ! 'selected is drawn from this table.',
; 0522 2 !
; 0523 2 ! BEGIN
; 0524 2 !
; 0525 2 ! UNIBUS_SIZER();
; 0526 2 !
; 0527 2 ! END;
; 0528 2 ! INCR N FROM 1 TO 3 DO
; 0529 2 ! BEGIN
```



```

: 0530 3
: 0531 3      DO                      ! Scope loop begins here
: 0532 4      BEGIN
: 0533 4
: 0534 4      UBI_CSR(.N) = PURGE;
: 0535 4
: 0536 4      CODECOMMENT
: 0537 4      ''
: 0538 4      'Set up the UET address register with the Unibus',
: 0539 4      'address. Bits 9 to 15 (the Unibus page frame) contain',
: 0540 4      'the 7 LSBs of the UBI map number. Bits 0 to 8 contain',
: 0541 4      'the byte number field of the CMI address.',
: 0542 4      ''
: 0543 5          BEGIN
: 0544 5
: 0545 5          SCRATCH[0] = 0;
: 0546 5          SCRATCH[1] = 0;
: 0547 5          UET_DATA_REG = XX'FF78';
: 0548 5          MAP_NUMBER = .FREE_MAP(.N);
: 0549 5          UBUS_PF = .MAP_NUMBER;
: 0550 5          UBUS_BO = SCRATCH[0];
: 0551 5          UET_ADDR_REG = .UBUS_ADDR;
: 0552 5
: 0553 4          END;
: 0554 4
: 0555 4      CODECOMMENT
: 0556 4      ''
: 0557 4      'Write the 2 MSBs of the Unibus address with the 2 MSBs of',
: 0558 4      'the map number.',
: 0559 4      ''
: 0560 5          BEGIN
: 0561 5
: 0562 5              TEMP3 = .MAP_NUMBER<7,2> * 8;
: 0563 5
: 0564 4          END;
: 0565 4
: 0566 4      CODECOMMENT
: 0567 4      ''
: 0568 4      'Set up the UBI map. Put the CMI page frame number (PFN)',
: 0569 4      'into the register BUF_PFN. Then call the map setup routine',
: 0570 4      'with the parameters map number, PFN of the CMI buffer,',
: 0571 4      'address offset enabled, and data path number.',
: 0572 4      ''
: 0573 5          BEGIN
: 0574 5
: 0575 5              TEMP = SCRATCH[0];
: 0576 5              BUF_PFN = .TEMP<9,15>;
: 0577 5              MAP_SETUP(.MAP_NUMBER,.BUF_PFN,1,.N);
: 0578 5
: 0579 4          END;
: 0580 4
: 0581 4      CODECOMMENT
: 0582 4      ''
: 0583 4      'Execute the first DATOB and after a nominal wait check that the',
: 0584 4      'two CMI locations are still empty.',

```

```

; 0585 4
; 0586 5
; 0587 5
; 0588 5
; 0589 5
; 0590 5
; L 0591 5
; xPRINT: Test 24, Subtest 0, Error 1
; L 0592 6
; xPRINT: Test 24, Subtest 0, Error 2
; 0593 5
; 0594 5
; 0595 6
; P 0596 6
; P 0597 6
; P 0598 6
; L 0599 6
; xPRINT: Test 24, Subtest 0, Error 3
; 0600 5
; 0601 5
; 0602 5
; 0603 6
; P 0604 6
; P 0605 6
; P 0606 6
; L 0607 6
; xPRINT: Test 24, Subtest 0, Error 4
; 0608 5
; 0609 5
; 0610 4
; 0611 4
; 0612 4
; 0613 4
; 0614 4
; 0615 4
; 0616 4
; 0617 5
; 0618 5
; 0619 5
; 0620 5
; 0621 5
; 0622 4
; 0623 4
; 0624 4
; 0625 4
; 0626 4
; 0627 4
; 0628 4
; 0629 5
; 0630 5
; 0631 5
; 0632 5
; 0633 5
; L 0634 5
; xPRINT:

      ``:
      BEGIN
          UET_CTRL_REG = .TEMP3 OR DATOB;

          $DS_WAITUS(TIME=10);
          UBI_STATUS(.N,UBIMOD_BDP);          ! M7
          UET_STATUS(UBIMOD_BDP);          ! M7
          IF .SCRATCH[0] NEQ 0
          THEN
              BEGIN
                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                              PRLINK = PRINT_GENERAL,
                              P1 = FMT_BDP_DATOB,P2 = 3,
                              P3 = .N,P4 = 0,P5 = .SCRATCH[0]);
                  Test 24, Subtest 0, Error 3
                  END;
                  IF .SCRATCH[1] NEQ 0
                  THEN
                      BEGIN
                          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
                                      PRLINK = PRINT_GENERAL,
                                      P1 = FMT_BDP_DATOB,P2 = 3,
                                      P3 = .N,P4 = 0,P5 = .SCRATCH[1]);
                          Test 24, Subtest 0, Error 4
                          END;
                      END;
                  CODECOMMENT
                  ``:
                  'Change the data in the UET data register and change',
                  'the Unibus address to the next sequential byte address.',
                  ``:
                  BEGIN
                      UET_DATA_REG = %X'56FF';
                      UET_ADDR_REG = .UBUS_ADDR + 1;
                  END;
                  CODECOMMENT
                  ``:
                  'Execute another DATOB and after a nominal wait check that the',
                  'CMI locations are still empty.',
                  ``:
                  BEGIN
                      UET_CTRL_REG = .TEMP3 OR DATOB;

                      $DS_WAITUS(TIME=10);
                      UBI_STATUS(.N,UBIMOD_BDP);          ! M7
                      Test 24, Subtest 0, Error 5

```



```
; L 0635 6      UET_STATUS(UBIMOD_BDP);      ! M7
; xPRINT:      Test 24, Subtest 0, Error 6
; 0636 5      IF .SCRATCH[0] NEQ 0
; 0637 5      THEN
; 0638 6          BEGIN
; P 0639 6          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
; P 0640 6          PRLINK = PRINT_GENERAL,
; P 0641 6          P1 = FMT_BDP_DATOB,P2 = 3,
; L 0642 6          P3 = .N,P4 = 0,P5 = .SCRATCH[0]);
; xPRINT:      Test 24, Subtest 0, Error 7
; 0643 5      END;
; 0644 5      IF .SCRATCH[1] NEQ 0
; 0645 5      THEN
; 0646 6          BEGIN
; P 0647 6          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
; P 0648 6          PRLINK = PRINT_GENERAL,
; P 0649 6          P1 = FMT_BDP_DATOB,P2 = 3,
; L 0650 6          P3 = .N,P4 = 0,P5 = .SCRATCH[1]);
; xPRINT:      Test 24, Subtest 0, Error 8
; 0651 5      END;
; 0652 5
; 0653 4      END;
; 0654 4
; 0655 4      CODECOMMENT
; 0656 4      '
; 0657 4      'Change the data in the UET data register and change',
; 0658 4      'the Unibus address to the next sequential byte address.',
; 0659 4      '
; 0660 5      BEGIN
; 0661 5
; 0662 5      UET_DATA_REG = xX'FF34';
; 0663 5      UET_ADDR_REG = .UBUS_ADDR + 2;
; 0664 5
; 0665 4      END;
; 0666 4
; 0667 4      CODECOMMENT
; 0668 4      '
; 0669 4      'Execute another DATOB and after a nominal wait check that the',
; 0670 4      'CMI locations are still empty.',
; 0671 4      '
; 0672 5      BEGIN
; 0673 5
; 0674 5      UET_CTRL_REG = .TEMP3 OR DATOB;
; 0675 5
; 0676 5      $DS_WAITUS(TIME=10);
; L 0677 5      UBI_STATUS(.N,UBIMOD_BDP);      ! M7
; xPRINT:      Test 24, Subtest 0, Error 9
; L 0678 6      UET_STATUS(UBIMOD_BDP);      ! M7
; xPRINT:      Test 24, Subtest 0, Error 10
; 0679 5      IF .SCRATCH[0] NEQ 0
; 0680 5      THEN
; 0681 6          BEGIN
; P 0682 6          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
; P 0683 6          PRLINK = PRINT_GENERAL,
; P 0684 6          P1 = FMT_BDP_DATOB,P2 = 3,
```



```
; L 0685 6
; xPRINT:
; 0686 5
; 0687 5
; 0688 5
; 0689 6
; P 0690 6
; P 0691 6
; P 0692 6
; L 0693 6
; xPRINT:
; 0694 5
; 0695 5
; 0696 4
; 0697 4
; 0698 4
; 0699 4
; 0700 4
; 0701 4
; 0702 4
; 0703 5
; 0704 5
; 0705 5
; 0706 5
; 0707 5
; 0708 4
; 0709 4
; 0710 4
; 0711 4
; 0712 4
; 0713 4
; 0714 4
; 0715 5
; 0716 5
; 0717 5
; 0718 5
; 0719 5
; L 0720 5
; xPRINT:
; L 0721 6
; xPRINT:
; 0722 5
; 0723 5
; 0724 6
; P 0725 6
; P 0726 6
; P 0727 6
; L 0728 6
; xPRINT:
; 0729 5
; 0730 5
; 0731 5
; 0732 6
; P 0733 6
; P 0734 6

P3 = .N,P4 = 0,P5 = .SCRATCH(0));
Test 24, Subtest 0, Error 11
END;
IF .SCRATCH(1) NEQ 0
THEN
BEGIN
$DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
PRLINK = PRINT_GENERAL,
P1 = FMT_BDP_DATOB,P2 = 3,
P3 = .N,P4 = 0,P5 = .SCRATCH(1));
Test 24, Subtest 0, Error 12
END;
END;
CODECOMMENT
''
'Change the data in the UET data register and change',
'the Unibus address to the next sequential byte address.',
''
BEGIN
UET_DATA_REG = %X'12FF';
UET_ADDR_REG = .UBUS_ADDR + 3;
END;
CODECOMMENT
''
'Execute another DATOB and after a nominal wait check that the',
'first CMI location now contains 34567800.',
''
BEGIN
UET_CTRL_REG = .TEMP3 OR DATOB;
$DS_WAITUS(TIME=10);
UBI_STATUS(.N,UBIMOD_BDP); ! M7
Test 24, Subtest 0, Error 13
UET_STATUS(UBIMOD_BDP); ! M7
Test 24, Subtest 0, Error 14
IF .SCRATCH(0) NEQ %X'34567800'
THEN
BEGIN
$DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
PRLINK = PRINT_GENERAL,
P1 = FMT_BDP_DATOB,P2 = 3,
P3 = .N,P4 = %X'34567800',P5 = .SCRATCH(0));
Test 24, Subtest 0, Error 15
END;
IF .SCRATCH(1) NEQ 0
THEN
BEGIN
$DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
PRLINK = PRINT_GENERAL,
```

```

; P 0735 6          P1 = FMT_BDP_DATOB,P2 = 3,
; L 0736 6          P3 = .N,P4 = 0,P5 = .SCRATCH[1]);
; xPRINT:          Test 24, Subtest 0, Error 16
; 0737 5          END;
; C738 5
; 0739 4          END;
; 0740 4
; 0741 4          CODECOMMENT
; 0742 4          'Now purge the BDP. This should cause the second CMI location to',
; 0743 4          'be written with the remaining byte.',
; 0744 4          '':
; 0745 4          BEGIN
; 0746 5
; 0747 5          UBI_CSR(.N) = PURGE;
; 0748 5          $DS_WAITUS(TIME=10);
; 0749 5          IF .SCRATCH[0] NEQ %X'34567800'
; 0750 5          THEN
; 0751 5              BEGIN
; 0752 6                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
; P 0753 6                      PRLINK = PRINT_GENERAL,
; P 0754 6                      P1 = FMT_BDP_DATOB,P2 = 3,
; P 0755 6                      P3 = .N,P4 = %X'34567800',P5 = .SCRATCH[0]);
; L 0756 6          Test 24, Subtest 0, Error 17
; xPRINT:          END;
; 0757 5          IF .SCRATCH[1] NEQ %X'00000012'
; 0758 5          THEN
; 0759 5              BEGIN
; 0760 6                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,
; P 0761 6                      PRLINK = PRINT_GENERAL,
; P 0762 6                      P1 = FMT_BDP_DATOB,P2 = 3,
; P 0763 6                      P3 = .N,P4 = %X'00000012',P5 = .SCRATCH[1]);
; L 0764 6          Test 24, Subtest 0, Error 18
; xPRINT:          END;
; 0765 5          END;
; 0766 5
; 0767 4          END;
; 0768 4
; 0769 4          END
; 0770 3          WHILE $DS_INLOOP;          ! Scope loop ends here
; 0771 3
; 0772 2          END;
; 0773 2
; 0774 1          $DS_ENDTEST;

```

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00018 \$:
00000000 00000003 00028

.ADDRESS P.AAD, P.AAE, P.AAF, TEST_024
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

44 42 20 74 65 73 66 66 4F 20 65 74 79 42 1A 00024 P.AAD: .WORD 24
00026 P.AAE: .ASCII <26>\Byte Offset BDP DATOB Test\

74 73 65 54 20 42 4F 54 41 44 20 50 00035
00041
00000000 00044 P.AAF: .BLKB 3
.LONG 0

.PSECT TEST_024,NOWRT,2

OFFC 00000

.ENTRY TEST_024, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0475
R11

5B 0000G CF 9E 00002
5A 0000' CF 9E 00007
59 00000000G 9F 9E 0000C

MOVAB UBIMOD_BDP, R11
MOVAB STATUS0, R10
MOVAB a#DS\$ERRHARD, R9

; Call the Unibus sizer routine to locate any Unibus memory. T
his builds
; a table of useable map entries. In the test which follows, t
he map
; selected is drawn from this table.

0000G CF 00 FB 00013
58 57 01 D0 00018
50 57 02 78 0001B 1\$:
58 0000G CF C1 0001F 2\$:
60 01 D0 00025

CALLS #0, UNIBUS_SIZER ; 0524
MOVL #1, N ; 0528
ASHL #2, N, R8 ; 0534
ADDL3 UBI_UNDER_TEST, R8, R0
MOVL #1, (R0)

; Set up the UET address register with the Unibus
address. Bits 9 to 15 (the Unibus page frame) contain
the 7 LSBs of the UBI map number. Bits 0 to 8 contain
the byte number field of the CMI address.

50 0000G CF 0000G CF 7C 00028
0003F462 8F C9 0002C
60 FF78 8F B0 00036
53 0000GCF47 3C 0003B
56 07 09 53 F0 00041
50 0000G CF 9E 00046
56 09 00 50 F0 0004B
50 0000G CF 0003F460 8F C9 00050
60 56 B0 0005A

CLRQ SCRATCH ; 0545
BISL3 #259170, UNIBUS_SPACE, R0 ; 0546
MOVW #-136, (R0) ; 0547
MOVZWL FREE_MAP[N], MAP_NUMBER ; 0548
INSV MAP_NUMBER, #9, #7, UBUS_ADDR ; 0549
MOVAB SCRATCH, R0 ; 0550
INSV R0, #0, #9, UBUS_ADDR
BISL3 #259168, UNIBUS_SPACE, R0
MOVW UBUS_ADDR, (R0) ; 0551

; Write the 2 MSBs of the Unibus address with the 2 MSBs of
the map number.

55 53 02 07 EF 0005D
55 08 C4 00062

EXTZV #7, #2, MAP_NUMBER, TEMP3 ; 0562
MULL2 #8, TEMP3

; Set up the UBI map. Put the CMI page frame number (PFN)
into the register BUF_PFN. Then call the map setup routine
with the parameters map number, PFN of the CMI buffer,
address offset enabled, and data path number.

52 54 54 0000G CF 9E 00065
OF 09 EF 0006A
57 DD 0006F

MOVAB SCRATCH, TEMP ; 0575
EXTZV #9, #15, TEMP, BUF_PFN ; 0576
PUSHL N ; 0577


```

01 DD 00071          PUSHL #1          ;
52 DD 00073          PUSHL BUF_PFN      ;
53 DD 00075          PUSHL MAP_NUMBER   ;
04 FB 00077          CALLS #4, MAP_SETUP ;

;
; Execute the first DATOB and after a nominal wait check that t
; he
; two CMI locations are still empty.
;

50      0000G CF 0003F464 8F C9 0007C          BISL3 #259172, UNIBUS_SPACE, R0          ; 0586
60      55      07 A9 00086          BISW3 #7, TEMP3, (R0)          ; 0588
7E      7E      0A 7D 0008A          MOVQ #10, -(SP)          ; 0590
04 AA 00000000G 9F      02 FB 0008D          CALLS #2, @DS$WAITUS          ;
AA 0000GDF47 1FFFFFFE 8F CB 00094          BICL3 #536870910, @UBI_UNDER_TEST[N], STATUS1 ; 0591
50      50      AA D0 000A0          MOVL STATUS1, R0          ;
1A 18 000A4          BGEQ 3$          ;
7E 7C 000A6          CLRQ -(SP)          ;
7E D4 000A8          CLRL -(SP)          ;
50 DD 000AA          PUSHL R0          ;
7E D4 000AC          CLRL -(SP)          ;
57 DD 000AE          PUSHL N          ;
0000G CF 9F 000B0          PUSHAB PRINT_CSR_ERR          ;
5B DD 000B4          PUSHL R11          ;
7E 0000G CF 9A 000B6          MOVZBL LUN, -(SP)          ;
01 DD 000BB          PUSHL #1          ;
0A FB 000BD          CALLS #10, DS$ERRHARD          ;
50 0000G CF 0003F464 8F C9 000C0 3$: BISL3 #259172, UNIBUS_SPACE, R0          ; 0592
6A 60 B0 000CA          MOVW (R0), STATUS0          ;
6A FF1E 8F AA 000CD          BICW2 #65310, STATUS0          ;
50 6A 32 000D2          CVTWL STATUS0, R0          ;
1B 13 000D5          BEQL 4$          ;
7E 7C 000D7          CLRQ -(SP)          ;
50 DD 000D9          PUSHL R0          ;
7E 02 7D 000DB          MOVQ #2, -(SP)          ;
0000G CF 9F 000DE          PUSHAB FMT_UET_STAT          ;
0000G CF 9F 000E2          PUSHAB PRINT_GENERAL          ;
5B DD 000E6          PUSHL R11          ;
7E 0000G CF 9A 000E8          MOVZBL LUN, -(SP)          ;
02 DD 000ED          PUSHL #2          ;
69 0A FB 000EF          CALLS #10, DS$ERRHARD          ;
50 0000G CF D0 000F2 4$: MOVL SCRATCH, R0          ; 0593
1E 13 000F7          BEQL 5$          ;
7E D4 000F9          CLRL -(SP)          ; 0599
50 DD 000FB          PUSHL R0          ;
7E D4 000FD          CLRL -(SP)          ;
57 DD 000FF          PUSHL N          ;
03 DD 00101          PUSHL #3          ;
0000G CF 9F 00103          PUSHAB FMT_BDP_DATOB          ;
0000G CF 9F 00107          PUSHAB PRINT_GENERAL          ;
5B DD 0010B          PUSHL R11          ;
7E 0000G CF 9A 0010D          MOVZBL LUN, -(SP)          ;
03 DD 00112          PUSHL #3          ;
69 0A FB 00114          CALLS #10, DS$ERRHARD          ;
50 0000G CF D0 00117 5$: MOVL SCRATCH+4, R0          ; 0601
1E 13 0011C          BEQL 6$          ;

```

```

;
; Change the data in the UET data register and change
; the Unibus address to the next sequential byte address.
;
50      0000G  CF 0003F462 8F C9 0013C 6$: BISL3 #259170, UNIBUS_SPACE, R0 ; 0617
60      56FF 8F B0 00146 MOVW #22271, (R0) ; 0619
50      0000G  CF 0003F460 8F C9 0014B BISL3 #259168, UNIBUS_SPACE, R0 ;
60      56 01 A1 00155 ADDW3 #1, UBUS_ADDR, (R0) ; 0620
;
; Execute another DATOB and after a nominal wait check that the
; CMI locations are still empty.
;
50      0000G  CF 0003F464 8F C9 00159 BISL3 #259172, UNIBUS_SPACE, R0 ; 0629
60      55 07 A9 00163 BISH3 #7, TEMP3, (R0) ; 0631
7E      0A 7D 00167 MOVQ #10, -(SP) ; 0633
04 AA 00000000G 9F 02 FB 0016A CALLS #2, a#DS$WAITUS ;
0000GDF47 1FFFFFFE 8F CB 00171 BICL3 #536870910, aUBI_UNDER_TEST[N], STATUS1 ; 0634
50      04 AA DO 0017D MOVL STATUS1, R0 ;
1A 18 00181 BGEQ 7$ ;
7E 7C 00183 CLRQ -(SP) ;
7E D4 00185 CLRQ -(SP) ;
50 DD 00187 PUSHL R0 ;
7E D4 00189 CLRQ -(SP) ;
57 DD 0018B PUSHL N ;
0000G CF 9F 0018D PUSHAB PRINT_CSR_ERR ;
5B DD 00191 PUSHL R11 ;
7E 0000G CF 9A 00193 MOVZBL LUN, -(SP) ;
05 DD 00198 PUSHL #5 ;
69 0A FB 0019A CALLS #10, DS$ERRHARD ;
50 0000G CF 0003F464 8F C9 0019D 7$: BISL3 #259172, UNIBUS_SPACE, R0 ; 0635
6A 60 B0 001A7 MOVW (R0), STATUS0 ;
6A FF1E 8F AA 001AA BICW2 #65310, STATUS0 ;
50 6A 32 001AF CVTWL STATUS0, R0 ;
1B 13 001B2 BEQL 8$ ;
7E 7C 001B4 CLRQ -(SP) ;
50 DD 001B6 PUSHL R0 ;
7E 02 7D 001B8 MOVQ #2, -(SP) ;
0000G CF 9F 001BB PUSHAB FMT_UET_STAT ;
0000G CF 9F 001BF PUSHAB PRINT_GENERAL ;
5B DD 001C3 PUSHL R11 ;
7E 0000G CF 9A 001C5 MOVZBL LUN, -(SP) ;
06 DD 001CA PUSHL #6 ;
69 0A FB 001CC CALLS #10, DS$ERRHARD ;
50 0000G CF D0 001CF 8$: MOVL SCRATCH, R0 ; 0636

```


				1E	13	001D4	BEQL	9\$	:	
				7E	D4	001D6	CLRL	-(SP)	:	0642
				50	DD	001D8	PUSHL	R0	:	
				7E	D4	001DA	CLRL	-(SP)	:	
				57	DD	001DC	PUSHL	N	:	
				03	DD	001DE	PUSHL	#3	:	
		0000G		CF	9F	001E0	PUSHAB	FMT_BDP_DATOB	:	
		0000G		CF	9F	001E4	PUSHAB	PRINT_GENERAL	:	
				5B	DD	001E8	PUSHL	R11	:	
	7E	0000G		CF	9A	001EA	MOVZBL	LUN, -(SP)	:	
				07	DD	001EF	PUSHL	#7	:	
	69			0A	FB	001F1	CALLS	#10, DS\$ERRHARD	:	
	50	0000G		CF	D0	001F4	9\$: MOVL	SCRATCH+4, R0	:	0644
				1E	13	001F9	BEQL	10\$	:	
				7E	D4	001FB	CLRL	-(SP)	:	0650
				50	DD	001FD	PUSHL	R0	:	
				7E	D4	001FF	CLRL	-(SP)	:	
				57	DD	00201	PUSHL	N	:	
				03	DD	00203	PUSHL	#3	:	
		0000G		CF	9F	00205	PUSHAB	FMT_BDP_DATOB	:	
		0000G		CF	9F	00209	PUSHAB	PRINT_GENERAL	:	
				5B	DD	0020D	PUSHL	R11	:	
	7E	0000G		CF	9A	0020F	MOVZBL	LUN, -(SP)	:	
				08	DD	00214	PUSHL	#8	:	
	69			0A	FB	00216	CALLS	#10, DS\$ERRHARD	:	
; Change the data in the UET data register and change										
; the Unibus address to the next sequential byte address.										
	50	0000G	CF	0003F462	8F	C9	00219	10\$: BISL3	#259170, UNIBUS_SPACE, R0	: 0660
			60	FF34	8F	B0	00223	MOVW	#-204, (R0)	: 0662
	50	0000G	CF	0003F460	8F	C9	00228	BISL3	#259168, UNIBUS_SPACE, R0	: 0663
	60		56		02	A1	00232	ADDW3	#2, UBUS_ADDR, (R0)	: 0663
; Execute another DATOB and after a nominal wait check that the										
; CMI locations are still empty.										
	50	0000G	CF	0003F464	8F	C9	00236	BISL3	#259172, UNIBUS_SPACE, R0	: 0672
	60		55		07	A9	00240	BISW3	#7, TEMP3, (R0)	: 0674
			7E		0A	7D	00244	MOVQ	#10, -(SP)	: 0676
		00000000G	9F		02	FB	00247	CALLS	#2, a#DS\$WAITUS	:
04	AA	0000G	D4	1FFFFFFE	8F	CB	0024E	BICL3	#536870910, aUPI_UNDER_TEST[N], STATUS1	: 0677
			50	04	AA	D0	0025A	MOVL	STATUS1, R0	:
					1A	18	0025E	BGEQ	11\$	:
					7E	7C	00260	CLRQ	-(SP)	:
					7E	D4	00262	CLRL	-(SP)	:
					50	DD	00264	PUSHL	R0	:
					7E	D4	00266	CLRL	-(SP)	:
					57	DD	00268	PUSHL	N	:
		0000G			CF	9F	0026A	PUSHAB	PRINT_CSR_ERR	:
					5B	DD	0026E	PUSHL	R11	:
	7E	0000G			CF	9A	00270	MOVZBL	LUN, -(SP)	:
					09	DD	00275	PUSHL	#9	:
					69	0A	FB	00277	CALLS	#10, DS\$ERRHARD
50	0000G	CF	0003F464	8F	C9	0027A	11\$: BISL3	#259172, UNIBUS_SPACE, R0	:	0678

6A		60	B0	00284	MOVW	(R0), STATUS0	:	
6A	FF1E	8F	AA	00287	BICW2	#65310, STATUS0	:	
50		6A	32	0028C	CVTL	STATUS0, R0	:	
		1B	13	0028F	BEQL	12\$	:	
		7E	7C	00291	CLRQ	-(SP)	:	
		50	DD	00293	PUSHL	R0	:	
7E		02	7D	00295	MOVQ	#2, -(SP)	:	
	0000G	CF	9F	00298	PUSHAB	FMT_UET_STAT	:	
	0000G	CF	9F	0029C	PUSHAB	PRINT_GENERAL	:	
		5B	DD	002A0	PUSHL	R11	:	
7E	0000G	CF	9A	002A2	MOVZBL	LUN, -(SP)	:	
		0A	DD	002A7	PUSHL	#10	:	
69		0A	FB	002A9	CALLS	#10, DS\$ERRHARD	:	
50	0000G	CF	D0	002AC	12\$:	MOVL	SCRATCH, R0	: 0679
		1E	13	002B1	BEQL	13\$	:	
		7E	D4	002B3	CLRL	-(SP)	: 0685	
		50	DD	002B5	PUSHL	R0	:	
		7E	D4	002B7	CLRL	-(SP)	:	
		57	DD	002B9	PUSHL	N	:	
		03	DD	002BB	PUSHL	#3	:	
	0000G	CF	9F	002BD	PUSHAB	FMT_BDP_DATOB	:	
	0000G	CF	9F	002C1	PUSHAB	PRINT_GENERAL	:	
		5B	DD	002C5	PUSHL	R11	:	
7E	0000G	CF	9A	002C7	MOVZBL	LUN, -(SP)	:	
		0B	DD	002CC	PUSHL	#11	:	
69		0A	FB	002CE	CALLS	#10, DS\$ERRHARD	:	
50	0000G	CF	D0	002D1	13\$:	MOVL	SCRATCH+4, R0	: 0687
		1E	13	002D6	BEQL	14\$	:	
		7E	D4	002D8	CLRL	-(SP)	: 0693	
		50	DD	002DA	PUSHL	R0	:	
		7E	D4	002DC	CLRL	-(SP)	:	
		57	DD	002DE	PUSHL	N	:	
		03	DD	002E0	PUSHL	#3	:	
	0000G	CF	9F	002E2	PUSHAB	FMT_BDP_DATOB	:	
	0000G	CF	9F	002E6	PUSHAB	PRINT_GENERAL	:	
		5B	DD	002EA	PUSHL	R11	:	
7E	0000G	CF	9A	002EC	MOVZBL	LUN, -(SP)	:	
		0C	DD	002F1	PUSHL	#12	:	
69		0A	FB	002F3	CALLS	#10, DS\$ERRHARD	:	
; Change the data in the UET data register and change								
; the Unibus address to the next sequential byte address.								
50	0000G	CF	0003F462	8F	C9	002F6	14\$: BISL3 #259170, UNIBUS_SPACE, R0	: 0703
		60	12FF	8F	B0	00300	MOVW #4863, (R0)	: 0705
50	0000G	CF	0003F460	8F	C9	00305	BISL3 #259168, UNIBUS_SPACE, R0	:
60		56		03	A1	0030F	ADDW3 #3, UBUS_ADDR, (R0)	: 0706
; Execute another DATOB and after a nominal wait check that the								
; first CMI location now contains 34567800.								
50	0000G	CF	0003F464	8F	C9	00313	BISL3 #259172, UNIBUS_SPACE, R0	: 0715
60		55		07	A9	0031D	BISW3 #7, TEMP3, (R0)	: 0717
		7E		0A	7D	00321	MOVQ #10, -(SP)	: 0719
	00000000G	9F		02	FB	00324	CALLS #2, a#DS\$WAITUS	:

04	AA	0000GDF47	1FFFFFFE	8F	CB	0032B	BICL3	#536870910, aUBI_UNDER_TEST[N], STATUS1	; 0720
		50	04	AA	D0	00337	MOVL	STATUS1, R0	:
				1A	18	0033B	BGEQ	15\$	:
				7E	7C	0033D	CLRQ	-(SP)	:
				7E	D4	0033F	CLRL	-(SP)	:
				50	DD	00341	PUSHL	R0	:
				7E	D4	00343	CLRL	-(SP)	:
				57	DD	00345	PUSHL	N	:
		0000G		CF	9F	00347	PUSHAB	PRINT_CSR_ERR	:
				5B	DD	0034B	PUSHL	R11	:
	7E	0000G		CF	9A	0034D	MOVZBL	LUN, -(SP)	:
				0D	DD	00352	PUSHL	#13	:
	69	0000G		0A	FB	00354	CALLS	#10, DS\$ERRHARD	:
50			0003F464	8F	C9	00357	BISL3	#259172, UNIBUS_SPACE, R0	; 0721
	6A			60	B0	00361	MOVW	(R0), STATUS0	:
	6A	FF1E		8F	AA	00364	BICW2	#65310, STATUS0	:
	50			6A	32	00369	CVTWL	STATUS0, R0	:
				1B	13	0036C	BEQL	16\$	:
				7E	7C	0036E	CLRQ	-(SP)	:
				50	DD	00370	PUSHL	R0	:
	7E			02	7D	00372	MOVQ	#2, -(SP)	:
		0000G		CF	9F	00375	PUSHAB	FMT_UET_STAT	:
		0000G		CF	9F	00379	PUSHAB	PRINT_GENERAL	:
				5B	DD	0037D	PUSHL	R11	:
	7E	0000G		CF	9A	0037F	MOVZBL	LUN, -(SP)	:
				0E	DD	00384	PUSHL	#14	:
	69			0A	FB	00386	CALLS	#10, DS\$ERRHARD	:
34567800	8F	0000G		CF	D1	00389	CMPL	SCRATCH, #878082048	; 0722
				24	13	00392	BEQL	17\$	:
				7E	D4	00394	CLRL	-(SP)	; 0728
		0000G		CF	DD	00396	PUSHL	SCRATCH	:
		34567800		8F	DD	0039A	PUSHL	#878082048	:
				57	DD	003A0	PUSHL	N	:
				03	DD	003A2	PUSHL	#3	:
		0000G		CF	9F	003A4	PUSHAB	FMT_BDP_DATOB	:
		0000G		CF	9F	003A8	PUSHAB	PRINT_GENERAL	:
				5B	DD	003AC	PUSHL	R11	:
	7E	0000G		CF	9A	003AE	MOVZBL	LUN, -(SP)	:
				0F	DD	003B3	PUSHL	#15	:
	69			0A	FB	003B5	CALLS	#10, DS\$ERRHARD	:
50		0000G		CF	D0	003B8	MOVL	SCRATCH+4, R0	; 0730
				1E	13	003BD	BEQL	18\$	:
				7E	D4	003BF	CLRL	-(SP)	; 0736
				50	DD	003C1	PUSHL	R0	:
				7E	D4	003C3	CLRL	-(SP)	:
				57	DD	003C5	PUSHL	N	:
				03	DD	003C7	PUSHL	#3	:
		0000G		CF	9F	003C9	PUSHAB	FMT_BDP_DATOB	:
		0000G		CF	9F	003CD	PUSHAB	PRINT_GENERAL	:
				5B	DD	003D1	PUSHL	R11	:
	7E	0000G		CF	9A	003D3	MOVZBL	LUN, -(SP)	:
				10	DD	003D8	PUSHL	#16	:
69				0A	FB	003DA	CALLS	#10, DS\$ERRHARD	:

; Now purge the BDP. This should cause the second CMI location

ZZ-ECCBA-1.4
UBI_TESTS_23_26
01-04

TEST_024: Byte Offset BDP DATOB Test
TEST_024: Byte Offset BDP DATOB Test

K 16

6-Sep-1985

Fiche 2 Frame K16

Sequence 411

6-Sep-1985 17:24:39

VAX-11 Bliss-32 V4.1-746

Page 28

6-Sep-1985 17:15:04

[LEMIEUX.ECCBA.RELEASE]ECCBA7.B32;5

(4)

;

to
; be written with the remaining byte.

58	57	02	78	003DD	18\$:	ASHL	#2, N, R8	:	0748
50	58	0000G	CF	C1 003E1		ADDL3	UBI_UNDER_TEST, R8, R0	:	
	60		01	D0 003E7		MOVL	#1, (R0)	:	
	7E		0A	7D 003EA		MOVQ	#10, -(SP)	:	0749
00000000G	9F		02	FB 003ED		CALLS	#2, a#DS\$WAITUS	:	
34567800	8F	0000G	CF	D1 003F4		CMPL	SCRATCH, #878082048	:	0750
			24	13 003FD		BEQL	19\$	:	
			7E	D4 003FF		CLRL	-(SP)	:	0756
		0000G	CF	DD 00401		PUSHL	SCRATCH	:	
	34567800		8F	DD 00405		PUSHL	#878082048	:	
			57	DD 0040B		PUSHL	N	:	
			03	DD 0040D		PUSHL	#3	:	
		0000G	CF	9F 0040F		PUSHAB	FMT_BDP_DATOB	:	
		0000G	CF	9F 00413		PUSHAB	PRINT_GENERAL	:	
			5B	DD 00417		PUSHL	R11	:	
	7E	0000G	CF	9A 00419		MOVZBL	LUN, -(SP)	:	
			11	DD 0041E		PUSHL	#17	:	
	69		0A	FB 00420		CALLS	#10, DS\$ERRHARD	:	
	12	0000G	CF	D1 00423	19\$:	CMPL	SCRATCH+4, #18	:	0758
			20	13 00428		BEQL	20\$	:	
			7E	D4 0042A		CLRL	-(SP)	:	0764
		0000G	CF	DD 0042C		PUSHL	SCRATCH+4	:	
			12	DD 00430		PUSHL	#18	:	
			57	DD 00432		PUSHL	N	:	
			03	DD 00434		PUSHL	#3	:	
		0000G	CF	9F 00436		PUSHAB	FMT_BDP_DATOB	:	
		0000G	CF	9F 0043A		PUSHAB	PRINT_GENERAL	:	
			5B	DD 0043E		PUSHL	R11	:	
	7E	0000G	CF	9A 00440		MOVZBL	LUN, -(SP)	:	
			12	DD 00445		PUSHL	#18	:	
	69		0A	FB 00447		CALLS	#10, DS\$ERRHARD	:	
00000000G	9F		00	FB 0044A	20\$:	CALLS	#0, a#DS\$INLOOP	:	0770
	03		50	E9 00451		BLBC	R0, 21\$	:	
		FBC8	31	00454		BRW	2\$	:	
FBBE	57		03	F1 00457	21\$:	ACBL	#3, #1, N, 1\$	:	0528
	00000000G		00	FB 0045D		CALLS	#0, a#DS\$BREAK	:	0772
			01	D0 00464		MOVL	#1, R0	:	0774
			04	00467		RET		:	

; Routine Size: 1128 bytes, Routine Base: TEST_024 + 0000


```
; 0775 2 $DS_BGNTST (SECTION=(DEFAULT,ALL),TEXT='Map Entry Functional Test');
; 0776 2
; 0777 2 !**
; 0778 2 ! TEST DESCRIPTION:
; 0779 2 !
; 0780 2 ! This tests the ability of the UBI to correctly handle transactions
; 0781 2 ! using all useable map registers. 496 map registers are potentially
; 0782 2 ! useable, but of that set some may not be useable due to the presence
; 0783 2 ! of Unibus memory. Therefore, before the testing of the maps is
; 0784 2 ! started, the Unibus is sized for Unibus memory to ascertain which of
; 0785 2 ! the maps are useable. A bitvector is generated by the Unibus sizer
; 0786 2 ! routine. If a map is useable, its bit in the bitvector is set.
; 0787 2 !
; 0788 2 ! For each map that is available for testing, a DATI with a unique
; 0789 2 ! pattern is set up, executed, and verified.
; 0790 2
; 0791 2 ! ASSUMPTIONS:
; 0792 2 !
; 0793 2 ! Test 1-Test 15
; 0794 2 !--
; 0795 2
; 0796 2 REGISTER
; 0797 2 BUF_PFN,
; 0798 2 MAP_ENTRY,
; 0799 2 N,
; 0800 2 UBUS_ADDR: WORD,
; 0801 2 TEMP1,
; 0802 2 TEMP3;
; 0803 2
; 0804 2 MACRO
; 0805 2 V_BIT = MAP_ENTRY<31,1>x, ! Defines the UBI map valid bit field ! M7
; 0806 2 UBUS_PF = UBUS_ADDR<9,7>x, ! Defines the Unibus page frame field
; 0807 2 UBUS_BO = UBUS_ADDR<0,9>x; ! Defines the Unibus byte number field
; 0808 2
; 0809 2 CODECOMMENT
; 0810 2 ''
; 0811 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
; 0812 2 'a bit map of useable map entries. In the test which follows, the maps',
; 0813 2 'to be used are checked for validity against this bit map.',
; 0814 2 ''
; 0815 2
; 0816 3 BEGIN
; 0817 3
; 0818 3 UNIBUS_SIZER();
; 0819 3
; 0820 3 END;
; 0821 2
; 0822 2 CODECOMMENT
; 0823 2 ''
; 0824 2 'Invalidate all map entries and initialize the UET data register.',
; 0825 2 ''
; 0826 2
; 0827 3 BEGIN
; 0828 3
; 0829 3 UET_DATA_REG = -1;
```

[illegible]


```

; 0885 6          END;
; 0886 6
; 0887 6          CODECOMMENT
; 0888 6          ''
; 0889 6          'Set up the UBI map. Put the CMI page frame number (PFN)',
; 0890 6          'into the register BUF_PFN. Then call the map setup routine',
; 0891 6          'with the parameters map number, PFN of the CMI buffer, byte',
; 0892 6          'offset mode (0 = no address offset), and data path number',
; 0893 6          '(0 is DDP).',
; 0894 6          '';
; 0895 7          BEGIN
; 0896 7
; 0897 7          TEMP1 = SCRATCH;
; 0898 7          BUF_PFN = .TEMP1<9,15>;
; 0899 7          MAP_SETUP(.N,.BUF_PFN,0,0);
; 0900 7
; 0901 6          END;
; 0902 6
; 0903 6          CODECOMMENT
; 0904 6          ''
; 0905 6          'Execute the DATI and after a nominal wait compare the',
; 0906 6          'contents of the UET data register with the expected',
; 0907 6          'value.',
; 0908 6          '';
; 0909 7          BEGIN
; 0910 7
; 0911 7          UET_CTRL_REG = .TEMP3 OR DATI;
; 0912 7
; 0913 7          $DS_WAITUS(TIME=10);
; L 0914 8          UET_STATUS(UBIMOD_MAP);          ! M7
; xPRINT:          Test 25, Subtest 0, Error 1
; 0915 7          TEMP1 = .UET_DATA_REG;
; 0916 7          IF .TEMP1 NEQ .N
; 0917 7          THEN
; 0918 8          BEGIN
; P 0919 8          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_MAP,PRLINK=PRINT_MAP_ERR,!M7 M13
; P 0920 8          P1=.N,P2=UBI_MAP(.N),P3=FMT_MAP_FAIL,          ! A13
; L 0921 8          P4=.N,P5=.TEMP1);          ! A13
; xPRINT:          Test 25, Subtest 0, Error 2
; 0922 8          ! $DS_PRINTX(FMT_MAP_FAIL,.N,.TEMP1);          ! A13
; 0923 7          END;
; 0924 7          UBI_MAP(.N) = .MAP_ENTRY;
; 0925 7
; 0926 6          END;
; 0927 6
; 0928 6          END
; 0929 5          WHILE $DS_INLOOP;          ! Scope loop ends here
; 0930 5
; 0931 4          END;
; 0932 4
; 0933 3          END;
; 0934 3
; 0935 2          END;
; 0936 2
; 0937 1          $DS_ENDTEST;

```



```

                                .PSECT DISPATCH,NOWRT,NOEXE,2
                                .ADDRESS P.AAG, P.AAH, P.AAI, TEST_025
                                .LONG 3, 0
                                .PSECT $PLIT$,NOWRT,NOEXE,2
00000000' 00000000' 00000000' 00000000' 00030 $:
                                00000000 00000003 00040
                                .PSECT $PLIT$,NOWRT,NOEXE,2
63 6E 75 46 20 79 72 74 6E 45 20 70 61 4D 19 00048 P.AAG: .WORD 25
74 73 65 54 20 6C 61 6E 6F 69 74 0004A P.AAH: .ASCII <25>\Map Entry Functional Test\
                                00059
                                00000000 00064 P.AAI: .LONG 0
                                .PSECT TEST_025,NOWRT,2
                                .ENTRY TEST_025, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0775
                                R11
                                MOVAB STATUS0, R11
                                MOVAB SCRATCH, R10
                                MOVAB @DS$ERRHARD, R9
                                MOVAB UNIBUS_SPACE, R8
; Call the Unibus sizer routine to locate any Unibus memory. T
; his builds
; a bit map of useable map entries. In the test which follows,
; the maps
; to be used are checked for validity against this bit map.
                                0000G CF 00 FB 00018 CALLS #0, UNIBUS_SIZER ; 0818
; Invalidate all map entries and initialize the UET data regis
; ter.
50 68 0003F462 8F C9 0001D BISL3 #259170, UNIBUS_SPACE, R0 ; 0827
60 01 AE 00025 MNEGW #1, (R0) ; 0829
53 01 1F 00 F0 00028 INSV #0, #31, #1, MAP_ENTRY ; 0830
50 D4 0002D CLRL N ; 0834
51 0000GDF40 DE 0002F 1$: MOVAL @UBI_UNDER_TEST[N], R1
0800 C1 53 D0 00035 MOVL MAP_ENTRY, 2048(R1)
ED 50 000001FF 8F F3 0003A AOBLEQ #511, N, 1$ ; 0831
; A map is testable if its bit is set in the bitvector VALID_MA
; PS. For maps
; 0 through 495 the following steps apply if the map is testabl
; e. Set up a
; DATI with a unique data pattern (actually, this is simply the
; map number).
; Execute and verify that the transaction was successful.
03 0000G CF 57 D4 00042 CLRL N ; 0849
57 E0 00044 2$: BBS N, VALID_MAPS, 3$ ; 0852

```

```

                                00DA 31 0004A      BRW      6$
                                ;
                                ; Set up the UET address register with the Unibus
                                ; address. Bits 9 to 15 (the Unibus page frame) contain the
                                ; 7 LSBs of the UBI map number. Bits 0 to 8 contain the byte
                                ; number field of the CMI address.
                                ;
                                3$:  MOVL      N, SCRATCH                      ; 0868
                                BISL3    #259170, UNIBUS_SPACE, R0          ;
                                CLRW      (R0)                              ; 0869
                                INSV      N, #9, #7, UBUS_ADDR              ; 0870
                                MOVAB     SCRATCH, R0                      ; 0871
                                INSV      R0, #0, #9, UBUS_ADDR              ;
                                BISL3    #259168, UNIBUS_SPACE, R0          ;
                                MOVW      UBUS_ADDR, (R0)                  ; 0872
                                ;
                                ; Write the two MSBs of the Unibus address with the two
                                ; MSBs of the map number.
                                ;
                                56      57      02      07      EF 00072      EXTZV    #7, #2, N, TEMP3      ; 0883
                                56      56      08      08      C4 00077      MULL2    #8, TEMP3          ;
                                ;
                                ; Set up the UBI map. Put the CMI page frame number (PFN)
                                ; into the register BUF_PFN. Then call the map setup routine
                                ; with the parameters map number, PFN of the CMI buffer, byte
                                ; offset mode (0 = no address offset), and data path number
                                ; (0 is DDP).
                                ;
                                52      55      55      6A      9E 0007A      MOVAB     SCRATCH, TEMP1      ; 0897
                                52      55      0F      09      EF 0007D      EXTZV    #9, #15, TEMP1, BUF_PFN ; 0898
                                7E      7C 00082      CLRQ      -(SP)          ; 0899
                                52      DD 00084      PUSHL     BUF_PFN          ;
                                57      DD 00086      PUSHL     N              ;
                                0000G CF      04      FB 00088      CALLS     #4, MAP_SETUP      ;
                                ;
                                ; Execute the DATI and after a nominal wait compare the
                                ; contents of the UET data register with the expected
                                ; value.
                                ;
                                50      68 0003F464 8F      C9 0008D      BISL3    #259172, UNIBUS_SPACE, R0 ; 0909
                                60      56      01      A9 00095      BISW3    #1, TEMP3, (R0)      ; 0911
                                7E      0A      7D 00099      MOVQ      #10, -(SP)      ; 0913
                                50      00000000G 9F      02      FB 0009C      CALLS     #2, @DS$WAITUS      ;
                                68      0003F464 8F      C9 000A3      BISL3    #259172, UNIBUS_SPACE, R0 ; 0914
                                6B      60      B0 000AB      MOVW      (R0), STATUS0      ;
                                6B      FF1E      8F      AA 000AE      BICW2    #65310, STATUS0      ;
                                50      6B      32 000B3      CVTWL    STATUS0, R0      ;
                                1D      13 000B6      BEQL      4$              ;
                                7E      7C 000B8      CLRQ      -(SP)          ;
                                50      DD 000BA      PUSHL     R0              ;
                                7E      02      7D 000BC      MOVQ      #2, -(SP)      ;
                                0000G CF      9F 000BF      PUSHAB   FMT_UET_STAT      ;
                                0000G CF      9F 000C3      PUSHAB   PRINT_GENERAL      ;
                                0000G CF      9F 000C7      PUSHAB   UBIMOD_MAP      ;
                                7E      0000G CF      9A 000CB      MOVZBL   LUN, -(SP)      ;

```


ZZ-ECCBA-1.4
UBI_TESTS_23_26
01-04

TEST_025: Map Entry Functional Test
TEST_025: Map Entry Functional Test

F 1

6-Sep-1985

Fiche 3 Frame F1

Sequence 417

6-Sep-1985 17:24:39

VAX-11 Bliss-32 V4.1-746

Page 34

6-Sep-1985 17:15:04

[LEMIEUX.ECCBA.RELEASE]ECCBA7.B32;5

(5)

		01	DD	000D0	PUSHL	#1	:			
		0A	FB	000D2	CALLS	#10, DS\$ERRHARD	:			
50	68	0003F462	8F	C9	000D5	4\$: BISL3	#259170, UNIBUS_SPACE, R0	: 0915		
	55		60	3C	000DD	MOVZWL	(R0), TEMP1	:		
	57		55	D1	000E0	CMPL	TEMP1, N	: 0916		
			2A	13	000E3	BEQL	5\$	:		
			7E	D4	000E5	CLRL	-(SP)	: 0921		
			55	DD	000E7	PUSHL	TEMP1	:		
			57	DD	000E9	PUSHL	N	:		
		0000G	CF	9F	000EB	PUSHAB	FMT_MAP_FAIL	:		
		0000G	DF	47	000EF	PUSHAL	aUBI_UNDER_TEST[N]	:		
	6E	00000800	8F	C0	000F4	ADDL2	#2048, (SP)	:		
			57	DD	000FB	PUSHL	N	:		
		0000G	CF	9F	000FD	PUSHAB	PRINT_MAP_ERR	:		
		0000G	CF	9F	00101	PUSHAB	UBIMOD_MAP	:		
	7E	0000G	CF	9A	00105	MOVZBL	LUN, -(SP)	:		
			02	DD	0010A	PUSHL	#2	:		
	69		0A	FB	0010C	CALLS	#10, DS\$ERRHARD	:		
	50	0000G	DF	47	DE	0010F	5\$: MOVAL	aUBI_UNDER_TEST[N], R0	: 0924	
		0800	C0		53	D0	00115	MOVL	MAP_ENTRY, 2048(R0)	:
	00000000G		9F		00	FB	0011A	CALLS	#0, a#DS\$INLOOP	: 0929
			03		50	E9	00121	BLBC	R0, 6\$	:
			FF	26	31	00124	BRW	3\$	:	
FF13	57	00000000G	01	000001EF	8F	F1	00127	6\$: ACBL	#495, #1, N, 2\$	: 0849
			9F		00	FB	00131	CALLS	#0, a#DS\$BREAK	: 0935
			50		01	D0	00138	MOVL	#1, R0	: 0937
					04	0013B	RET	:	:	:

; Routine Size: 316 bytes, Routine Base: TEST_025 + 0000


```
; 0938 2 $DS_BGNTTEST (SECTION=(DEFAULT,ALL),TEXT='CSR Status Bit Test');
; 0939 2
; 0940 2 !++
; 0941 2 ! TEST DESCRIPTION:
; 0942 2 !
; 0943 2 ! This tests that there are no stuck at zero faults in each of the
; 0944 2 ! three CSRs. The diagnostic features of the memory controller are
; 0945 2 ! used to simulate the uncorrectable error status bit. To test the
; 0946 2 ! NXM bit, an illegal nexus address is used.
; 0947 2 !
; 0948 2 ! The first subtest is concerned with verifying the functionality of
; 0949 2 ! the NXM bit. The UBI and UET are set up to execute
; 0950 2 ! a DATI from a non-existent CMI nexus (F40000 for now). The DATI is
; 0951 2 ! initiated, and after a nominal wait, the CSR is read, checking that
; 0952 2 ! both NXM and ERR are set. This is repeated for each BDP.
; 0953 2 !
; 0954 2 ! The second subtest is concerned with verifying the functionality of
; 0955 2 ! the UCE bit. A data pattern is written into memory with ECC disabled.
; 0956 2 ! This causes the check bits for this pattern to be written into the
; 0957 2 ! memory controller CSR 1 <06:00>. ECC is enabled, and a second pattern
; 0958 2 ! is written to the same location, simulating a double-bit error. The
; 0959 2 ! UBI and UET are set up for a DATI from this memory
; 0960 2 ! location. Then the memory controller is put into Diagnostic Check
; 0961 2 ! Mode and Page Mode. This will cause the check bits to be drawn from
; 0962 2 ! the memory controller CSR 1 <06:00> instead of memory, so that an
; 0963 2 ! uncorrectable error will be indicated when the location in question is
; 0964 2 ! read. A DATI is initiated, which should cause an uncorrectable error
; 0965 2 ! to be indicated by the memory controller. After a nominal wait, the
; 0966 2 ! UBI CSR is read, checking that both UCE and ERR are set. This is
; 0967 2 ! repeated for each BDP. The PE bit in the UET is also checked.
; 0968 2 !
; 0969 2 ! ASSUMPTIONS:
; 0970 2 !
; 0971 2 ! Test 1-Test 15
; 0972 2 !--
; 0973 2
; 0974 2 LOCAL
; 0975 2     BUF_PFN,
; 0976 2     MAP_ENTRY,
; 0977 2     MAP_NUMBER,
; 0978 2     M,
; 0979 2     MEM_CSR: REF VECTOR[3],
; 0980 2     N,
; 0981 2     UBUS_ADDR: WORD,
; 0982 2     TEMP1,
; 0983 2     TEMP2,
; 0984 2     TEMP3,
; 0985 2     TEMP4;
; 0986 2
; 0987 2 MACRO
; 0988 2     V_BIT = MAP_ENTRY<31,1>%;
; 0989 2     UBUS_PF = UBUS_ADDR<9,7>%;
; 0990 2     UBUS_BO = UBUS_ADDR<0,9>%;
; 0991 2
; 0992 2
```

! Defines the UBI map valid bit field ! M7
! Defines the Unibus page frame field
! Defines the Unibus byte number field

```

: 0993 2 ! $DS_SETIPL(LEVEL=10); ! D8
: 0994 2
: 0995 2
: 0996 2 CODECOMMENT
: 0997 2 ''
: 0998 2 'Call the Unibus sizer routine to locate any Unibus memory. This builds',
: 0999 2 'a bit map of useable map entries. In the test which follows, the maps',
: 1000 2 'to be used are checked for validity against this bit map.',
: 1001 2 ''
: 1002 2
: 1003 3 BEGIN
: 1004 3
: 1005 3 UNIBUS_SIZER();
: 1006 3 MAP_NUMBER = .FREE_MAP[0];
: 1007 3
: 1008 2 END;
: 1009 2
: 1010 3 $DS_BGNSUB; ! Subtest 1: NXM
: 1011 3
: 1012 3 CODECOMMENT
: 1013 3 ''
: 1014 3 'Set up a DATI from a non-existent CMI nexus. Execute the DATI and read the',
: 1015 3 'the CSR, verifying that both the NXM and ERR bits are set. Repeat this for',
: 1016 3 'all three BDPs.',
: 1017 3 ''
: 1018 4 BEGIN
: 1019 4
: 1020 4 SCRATCH = NON_XIST_NEXUS;
: 1021 4 INCR N FROM 1 TO 3 DO
: 1022 5 BEGIN
: 1023 5
: 1024 5 DO ! Scope loop begins here
: 1025 6 BEGIN
: 1026 6
: 1027 6 UBI_CSR(.N) = PURGE; ! Purge data path
: 1028 6
: 1029 6 CODECOMMENT
: 1030 6 ''
: 1031 6 'Set up the UET address register with the Unibus',
: 1032 6 'address. Bits 9 to 15 (the Unibus page frame) contain the',
: 1033 6 '7 LSBs of the UBI map number. Bits 0 to 8 contain the byte',
: 1034 6 'numbr field of the CMI address.',
: 1035 6 ''
: 1036 7 BEGIN
: 1037 7
: 1038 7 UBUS_PF = .MAP_NUMBER;
: 1039 7 UBUS_BO = .SCRATCH;
: 1040 7 UET_ADDR_REG = .UBUS_ADDR;
: 1041 7
: 1042 6 END;
: 1043 6
: 1044 6 CODECOMMENT
: 1045 6 ''
: 1046 6 'Write the two MSBs of the Unibus address with the two',
: 1047 6 'MSBs of the map number. Make sure the NXM bit is clear.',

```



```

; 1048 6      ``:
; 1049 7      BEGIN
; 1050 7
; 1051 7      UBI_CSR(.N) = NXM;      ! Write one to clear
; 1052 7
; 1053 7      TEMP4 = .MAP_NUMBER<7,2> * 8;
; 1054 7
; 1055 6      END;
; 1056 6
; 1057 6      CODECOMMENT
; 1058 6      ``:
; 1059 6      'Set up the UBI map. Put the CMI page frame number (PFN)',
; 1060 6      'into the register BUF_PFN. Then call the map setup routine',
; 1061 6      'with the parameters map number, PFN of the CMI buffer, byte',
; 1062 6      'offset mode (0 = no address offset), and data path number.',
; 1063 6      ``:
; 1064 7      BEGIN
; 1065 7
; 1066 7      TEMP1 = .SCRATCH;
; 1067 7      BUF_PFN = .TEMP1<9,15>;
; 1068 7      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.N);
; 1069 7
; 1070 6      END;
; 1071 6
; 1072 6      CODECOMMENT
; 1073 6      ``:
; 1074 6      'Execute the DATI and after a nominal wait read the CSR,',
; 1075 6      'verifying that the NXM and ERR bits are set.',
; 1076 6      ``:
; 1077 7      BEGIN
; 1078 7
; 1079 7      UET_CTRL_REG = .TEMP4 OR DATI;
; 1080 7
; 1081 7      $DS_WAITMS(TIME=1,RETTIM=0);
; 1082 7      TEMP1 = .UBI_CSR(.N) AND CSR_MASK;
; 1083 7      TEMP2 = (NXM OR ERR) AND CSR_MASK;
; 1084 7      IF .TEMP1 NEQ .TEMP2
; 1085 7      THEN
; 1086 8          BEGIN
; 1087 8              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; 1088 8                  PRLINK=PRINT_CSR_ERR,
; 1089 8                  P1=.N,P2=.TEMP2,P3=.TEMP3);
; 1090 7
; 1091 6      END;
; 1092 6
; 1093 6      CODECOMMENT
; 1094 6      ``:
; 1095 6      'Check that the TO bit set in the UET.',
; 1096 6      ``:
; 1097 7      BEGIN
; 1098 7      TEMP1 = .UET_CTRL_REG; ! Read the control register
; 1099 7      TEMP1 = .TEMP1 AND %X'40';
; 1100 7      IF .TEMP1 EQL 0
; 1101 7      THEN

```

Test 26, Subtest 1, Error 1


```

; 1102 8          BEGIN
; P 1103 8          $DS_ERRHARD(UNIT=.LUN,MSGADR=UETMOD,
; P 1104 8          PRLINK = PRINT_GENERAL,
; P 1105 8          P1 = FMT_UET_STAT,P2 = 2,
; L 1106 8          P3 = %X'40',P4 = .TEMP1);
; %PRINT:          Test 26, Subtest 1, Error 2
; 1107 7          END;
; 1108 6          END;
; 1109 6
; 1110 6          CODECOMMENT
; 1111 6          'Check that the NXM bit clears in the UBI CSR.',
; 1112 6          '':
; 1113 6          BEGIN
; 1114 7          UBI_CSR(.N) = NXM;          ! Clear the NXM
; 1115 7          UBI_STATUS(.N,UBIMOD_CSR);    ! Check that it cleared ! A8
; L 1116 7          Test 26, Subtest 1, Error 3
; %PRINT:          END;
; 1117 6
; 1118 6          CODECOMMENT
; 1119 6          'Check that the TO bit clears with a UET INIT.',
; 1120 6          '':
; 1121 6          BEGIN
; 1122 6          UET_CTRL_REGB = %X'80';          ! Clear the UET Error
; 1123 7          TEMP1 = .UET_CTRL_REG AND %X'40';
; 1124 7          IF .TEMP1 NEQ 0
; 1125 7          THEN
; 1126 7          BEGIN
; 1127 7          $DS_ERRHARD(UNIT=.LUN,MSGADR=UETMOD,
; P 1128 8          PRLINK = PRINT_GENERAL,
; P 1129 8          P1 = FMT_UET_STAT,P2 = 2,
; P 1130 8          P3 = 0,P4 = .TEMP1);
; P 1131 8
; L 1132 8          Test 26, Subtest 1, Error 4
; %PRINT:          END;
; 1133 7          END;
; 1134 6          END;
; 1135 6
; 1136 6          END
; 1137 5          WHILE $DS_INLOOP;          ! Scope loop ends here
; 1138 5
; 1139 4          END;
; 1140 4
; 1141 3          END;
; 1142 3          $DS_ENDSUB;          ! End of Subtest 1
; 1143 2
; 1144 2          $DS_BGNSUB;          ! Subtest 2: UCE
; 1145 3
; 1146 3          INCR N FROM 1 TO 3 DO
; 1147 3          BEGIN
; 1148 4
; 1149 4          DO          ! Scope loop begins here
; 1150 4          BEGIN
; 1151 5          UBI_CSR(.N) = PURGE;
; 1152 5
; 1153 5

```

```
; 1154 5
; 1155 5
; 1156 5
; 1157 5
; 1158 5
; 1159 5
; 1160 5
; 1161 5
; 1162 6
; 1163 6
; 1164 6
; 1165 6
; 1166 6
; 1167 6
; 1168 6
; 1169 6
; 1170 6
; 1171 6
; 1172 6
; 1173 6
; 1174 5
; 1175 5
; 1176 5
; 1177 5
; 1178 5
; 1179 5
; 1180 5
; 1181 5
; 1182 5
; 1183 6
; 1184 6
; 1185 6
; 1186 6
; 1187 6
; 1188 6
; 1189 5
; 1190 5
; 1191 5
; 1192 5
; 1193 5
; 1194 5
; 1195 5
; 1196 6
; 1197 6
; 1198 6
; 1199 6
; 1200 5
; 1201 5
; 1202 5
; 1203 5
; 1204 5
; 1205 5
; 1206 5
; 1207 5
; 1208 5

CODECOMMENT
,,
'Turn on ECC Disable Mode in the memory controller, write a pattern',
'to location SCRATCH, then turn off ECC Disable Mode. Write a',
'second pattern to SCRATCH, with 2 bits different from the first',
'pattern.',
,,
BEGIN
MEM_CSR = MEMORY_CONT;
TEMP3 = DISABLE_ECC; ! D8
TEMP3 = SCRATCH; ! A8
MEM_CSR[1] = (.TEMP3<9,15> * 512) OR DISABLE_ECC OR PAGE_MODE;
! Turn on ECC Disable mode M8
SCRATCH = 3;
TEMP3 = NOT ENABLE_ECC; ! D8
MEM_CSR[1] = .MEM_CSR[1] AND (NOT DISABLE_ECC); ! Turn off ECC Disable mode M8
SCRATCH = 0;
END;

CODECOMMENT
,,
'The memory is now primed for an uncorrectable error. Set up the',
'UET address register with the Unibus address. Bits 9 to 15',
'(the Unibus page frame) contain the 7 LSBs of the UBI map number.',
'Bits 0 to 8 contain the byte number field of the CMI address.',
,,
BEGIN
UBUS_PF = .MAP_NUMBER;
UBUS_BO = SCRATCH;
UET_ADDR_REG = .UBUS_ADDR;
END;

CODECOMMENT
,,
'Write the two MSBs of the Unibus address with the two',
'MSBs of the map number.',
,,
BEGIN
TEMP4 = .MAP_NUMBER<7,2> * 8;
END;

CODECOMMENT
,,
'Set up the UBI map. Put the CMI page frame number (PFN)',
'into the register BUF_PFN. Then call the map setup routine',
'with the parameters map number, PFN of the CMI buffer, byte',
'offset mode (0 = no address offset), and data path number.',
,,
```

```

; 1209 6      BEGIN
; 1210 6
; 1211 6      TEMP1 = SCRATCH;
; 1212 6      BUF_PFN = .TEMP1<9,15>;
; 1213 6      MAP_SETUP(.MAP_NUMBER,.BUF_PFN,0,.N);
; 1214 6
; 1215 5      END;
; 1216 5
; 1217 5      CODECOMMENT
; 1218 5      ''
; 1219 5      'Now turn on the memory controllers Diagnostic Check Mode and Page',
; 1220 5      'Mode, and execute the DATI. After a nominal wait check that the',
; 1221 5      'UCE and ERR bits are set in the CSR.',
; 1222 5      ''
; 1223 6      BEGIN
; 1224 6
; 1225 6      !      TEMP3<9,15> = .BUF_PFN;                      ! D8
; 1226 6      !      TEMP3 = .TEMP3 OR DIAG_CHK_MODE;              ! D8
; 1227 6      !      MEM_CSR[1] = .MEM_CSR[1] OR DIAG_CHK_MODE;    ! M8
; 1228 6
; 1229 6      UET_CTRL_REG = .TEMP4 OR DATI;
; 1230 6
; 1231 6      $DS_WAITUS(TIME=10);
; 1232 6      MEM_CSR[1] = 0;                      ! A8
; 1233 6      MEM_CSR[0] = CLEAR_MEM_ERR;          ! A8
; 1234 6      TEMP1 = .UBI_CSR(.N) AND CSR_MASK;
; 1235 6      MEM_CSR[1] = 0;
; 1236 6      TEMP2 = (UCE OR ERR) AND CSR_MASK;
; 1237 6      IF .TEMP1 NEQ .TEMP2
; 1238 6      THEN
; 1239 7          BEGIN
; 1240 7          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; 1241 7          PRLINK=PRINT_CSR_ERR,
; 1242 7          P1=.N,P2=.TEMP2,P3=.TEMP1);
; 1243 6      %PRINT:      Test 26, Subtest 2, Error 5
; 1244 5      END;
; 1245 5      END;
; 1246 5      CODECOMMENT
; 1247 5      ''
; 1248 5      'Check that the PE bit set in the UET.',
; 1249 5      ''
; 1250 6      BEGIN
; 1251 6      TEMP1 = .UET_CTRL_REG;
; 1252 6      TEMP1 = .TEMP1 AND %X'80';
; 1253 6      IF .TEMP1 EQL 0
; 1254 6      THEN
; 1255 7          BEGIN
; 1256 7          $DS_ERRHARD(UNIT=.LUN,MSGADR=UETMOD,
; 1257 7          PRLINK = PRINT_GENERAL,
; 1258 7          P1 = FMT_UET_STAT,P2 = 2,
; 1259 7          P3 = %X'80',P4 = .TEMP1);
; 1260 6      %PRINT:      Test 26, Subtest 2, Error 6
; 1261 5      END;
; 1261 5      END;

```



```

; 1262 5
; 1263 5
; 1264 5
; 1265 5
; 1266 5
; 1267 6
; 1268 6
; L 1269 6
; *PRINT: Test 26, Subtest 2, Error 7
; 1270 5
; 1271 5
; 1272 5
; 1273 5
; 1274 5
; 1275 5
; 1276 6
; 1277 6
; 1278 6
; 1279 6
; 1280 6
; 1281 7
; P 1282 7
; P 1283 7
; P 1284 7
; L 1285 7
; *PRINT: Test 26, Subtest 2, Error 8
; 1286 6
; 1287 5
; 1288 5
; 1289 5
; 1290 4
; 1291 4
; 1292 3
; 1293 3
; 1294 2
; 1295 2
; 1296 1

CODECOMMENT
;
; 'Check that the UCE bit clears in the UBI CSR.',
;
; BEGIN
;   UBI_CSR(.N) = UCE;      ! Clear the UCE bit
;   UBI_STATUS(.N,UBIMOD_CSR); ! Check that it cleared      ! A8
; END;

CODECOMMENT
;
; 'Check that the PE bit clears with a UET INIT.',
;
; BEGIN
;   UET_CTRL_REGB = %X'80';
;   TEMP1 = UET_CTRL_REG AND %X'80';
;   IF TEMP1 NEQ 0
;   THEN
;     BEGIN
;       $DS_ERRHARD(UNIT=.LUN,MSGADR=UETMOD,
;                   PRLINK = PRINT_GENERAL,
;                   P1 = FMT_UET_STAT,P2 = 2,
;                   P3 = 0,P4 = TEMP1);
;     END;
;   END;
; END
; WHILE $DS_INLOOP;      ! Scope loop ends here
; END;
$DS_ENDSUB;              ! End of Subtest 2
$DS_ENDTEST;

```

```

.PSECT DISPATCH,NOWRT,NOEXE,2
00000000' 00000000' 00000000' 00000000' 00048 $: .ADDRESS P.AAJ, P.AAK, P.AAL, TEST_026
00000000 00000003 00058 .LONG 3, 0
.PSECT $PLIT$,NOWRT,NOEXE,2
001A 00068 P.AAJ: .WORD 26
74 69 42 20 73 75 74 61 74 53 20 52 53 43 13 0006A P.AAK: .ASCII <19>\CSR Status Bit Test\
74 73 65 54 20 00079
0007E .BLKB 2
00000000 00080 P.AAL: .LONG 0
.EXTRN DS$BGNSUB, DS$WAITMS
.EXTRN DS$ENDSUB

```

[illegible]

```

0000G CF 0440 8F BB 00069 PUSH R #^M<R6,R10>
04 FB 0006D CALLS #4, MAP_SETUP
;
; Execute the DATI and after a nominal wait read the CSR,
; verifying that the NXM and ERR bits are set.
;
50 0000G CF 0003F464 8F C9 00072 BISL3 #259172, UNIBUS_SPACE, R0 ; 1077
60 59 01 A9 0007C BISW3 #1, TEMP4, (R0) ; 1079
7E 01 7D 00080 MOVQ #1, -(SP) ; 1081
00000000G 9F 02 FB 00083 CALLS #2, a#DS$WAITMS ;
55 0000GDF43 1FFFFFFE 8F CB 0008A BICL3 #536870910, aUBI_UNDER_TEST[N], TEMP1 ; 1082
57 C0000000 8F D0 00095 MOVL #-1073741824, TEMP2 ; 1083
57 55 D1 0009C CMPL TEMP1, TEMP2 ; 1084
1C 13 0009F BEQL 2$ ;
7E 7C 000A1 CLRQ -(SP) ; 1089
7E D4 000A3 CLRL -(SP) ;
52 DD 000A5 PUSHL TEMP3 ;
8F BB 000A7 PUSHR #^M<R3,R7> ;
0000G CF 9F 000AB PUSHAB PRINT_CSR_ERR ;
0000G CF 9F 000AF PUSHAB UBIMOD_CSR ;
7E 0000G CF 9A 000B3 MOVZBL LUN, -(SP) ;
01 DD 000B8 PUSHL #1 ;
6B 0A FB 000BA CALLS #10, DS$ERRHARD ;
;
; Check that the TO bit set in the UET.
;
50 0000G CF 0003F464 8F C9 000BD 2$: BISL3 #259172, UNIBUS_SPACE, R0 ; 1098
55 60 3C 000C7 MOVZWL (R0), TEMP1 ;
55 FFFFFFFBF 8F CA 000CA BICL2 #-65, TEMP1 ; 1099
20 12 000D1 BNEQ 3$ ; 1100
7E 7C 000D3 CLRQ -(SP) ; 1106
55 DD 000D5 PUSHL TEMP1 ;
7E 40 8F 9A 000D7 MOVZBL #64, -(SP) ;
02 DD 000DB PUSHL #2 ;
0000G CF 9F 000DD PUSHAB FMT_UET_STAT ;
0000G CF 9F 000E1 PUSHAB PRINT_GENERAL ;
0000G CF 9F 000E5 PUSHAB UETMOD ;
7E 0000G CF 9A 000E9 MOVZBL LUN, -(SP) ;
02 DD 000EE PUSHL #2 ;
6B 0A FB 000F0 CALLS #10, DS$ERRHARD ;
;
; Check that the NXM bit clears in the UBI CSR.
;
0000' CF 0000GDF43 40000000 8F D0 000F3 3$: MOVL #1073741824, aUBI_UNDER_TEST[N] ; 1115
0000GDF43 1FFFFFFE 8F CB 000FD BICL3 #536870910, aUBI_UNDER_TEST[N], STATUS1 ; 1116
50 0000' CF D0 0010A MOVL STATUS1, R0 ;
1C 18 0010F BGEQ 4$ ;
7E 7C 00111 CLRQ -(SP) ;
7E D4 00113 CLRL -(SP) ;
50 DD 00115 PUSHL R0 ;
7E D4 00117 CLRL -(SP) ;
53 DD 00119 PUSHL N ;
0000G CF 9F 0011B PUSHAB PRINT_CSR_ERR ;
0000G CF 9F 0011F PUSHAB UBIMOD_CSR ;
7E 0000G CF 9A 00123 MOVZBL LUN, -(SP) ;

```



```

                                03 DD 00128      PUSHL #3
                                0A FB 0012A      CALLS #10, DS$ERRHARD
                                ;
                                ; Check that the TO bit clears with a UET INIT.
                                ;
50      0000G CF 0003F465      8F C9 0012D 4$: BISL3 #259173, UNIBUS_SPACE, R0      ; 1123
                                8F 90 00137      MOVB #-128, (R0)      ; 1124
50      0000G CF 0003F464      8F C9 0013B      BISL3 #259172, UNIBUS_SPACE, R0      ; 1125
                                60 3C 00145      MOVZWL (R0), TEMP1      ;
                                55 FFFFFFFBF      8F CA 00148      BICL2 #-65, TEMP1      ;
                                1D 13 0014F      BEQL 5$      ; 1126
                                7E 7C 00151      CLRQ -(SP)      ; 1132
                                55 DD 00153      PUSHL TEMP1      ;
                                02 7D 00155      MOVQ #2, -(SP)      ;
                                0000G CF 9F 00158      PUSHAB FMT_UET_STAT      ;
                                0000G CF 9F 0015C      PUSHAB PRINT_GENERAL      ;
                                0000G CF 9F 00160      PUSHAB UETMOD      ;
                                7E 0000G CF 9A 00164      MOVZBL LUN, -(SP)      ;
                                04 DD 00169      PUSHL #4      ;
                                6B 0A FB 0016B      CALLS #10, DS$ERRHARD      ;
00000000G 9F 00 FB 0016E 5$: CALLS #0, a#DS$INLOOP      ; 1137
                                03 50 E9 00175      BLBC R0, 7$      ;
                                FEAF 31 00178 6$: BRW 1$      ;
F9      53 03 F3 0017B 7$: AOBLEQ #3, N, 6$      ; 1021
                                01 DD 0017F      PUSHL #1      ; 1141
                                1A DD 00181      PUSHL #26      ;
00000000G 9F 02 FB 00183      CALLS #2, a#DS$ENDSUB      ;
                                02 DD 0018A      PUSHL #2      ; 1143
                                1A DD 0018C      PUSHL #26      ;
00000000G 9F 02 FB 0018E      CALLS #2, a#DS$BGNSUB      ;
                                54 01 D0 00195      MOVL #1, N      ; 1147
0000GDF44 01 D0 00198 8$: MOVL #1, aUBI_UNDER_TEST[N]      ; 1153
                                ;
                                ; Turn on ECC Disable Mode in the memory controller, write a pa
                                ; ttern
                                ; to location SCRATCH, then turn off ECC Disable Mode. Write a
                                ; second pattern to SCRATCH, with 2 bits different from the fir
                                ; st
                                ; pattern.
                                ;
                                53 00F20000      8F D0 0019E      MOVL #15859712, MEM_CSR      ; 1164
                                52 0000G CF 9E 001A5      MOVAB SCRATCH, TEMP3      ; 1166
00      0F 09 EF 001AA      EXTZV #9, #15, TEMP3, R0      ; 1167
                                50 09 78 001AF      ASHL #9, R0, R0      ;
                                04 50 0A000000      8F C9 001B3      BISL3 #167772160, R0, 4(MEM_CSR)      ;
                                0000G CF 03 D0 001BC      MOVL #3, SCRATCH      ; 1169
                                07 A3 02 8A 001C1      BICB2 #2, 7(MEM_CSR)      ; 1171
                                0000G CF D4 001C5      CLRL SCRATCH      ; 1172
                                ;
                                ; The memory is now primed for an uncorrectable error. Set up
                                ; the
                                ; UET address register with the Unibus address. Bits 9 to 15
                                ; (the Unibus page frame) contain the 7 LSBs of the UBI map num
                                ; ber.
                                ; Bits 0 to 8 contain the byte number field of the CMI address.

```

```

58          07          09          56 F0 001C9      ;      INSV      MAP_NUMBER, #9, #7, UBUS_ADDR      ; 1185
          50          50      0000G CF 9E 001CE      ;      MOVAB     SCRATCH, R0      ; 1186
58          09          00          50 F0 001D3      ;      INSV      R0, #0, #9, UBUS_ADDR      ;
          50      0000G CF 0003F460 8F C9 001D8      ;      BISL3     #259168, UNIBUS_SPACE, R0      ;
          60          58 B0 001E2      ;      MOVW      UBUS_ADDR, (R0)      ; 1187
;
; Write the two MSBs of the Unibus address with the two
; MSBs of the map number.
;
59          56          02          07 EF 001E5      ;      EXTZV     #7, #2, MAP_NUMBER, TEMP4      ; 1198
          59          08 C4 001EA      ;      MULL2     #8, TEMP4      ;
;
; Set up the UBI map. Put the CMI page frame number (PFN)
; into the register BUF_PFN. Then call the map setup routine
; with the parameters map number, PFN of the CMI buffer, byte
; offset mode (0 = no address offset), and data path number.
;
          55          55      0000G CF 9E 001ED      ;      MOVAB     SCRATCH, TEMP1      ; 1211
          0F          09 EF 001F2      ;      EXTZV     #9, #15, TEMP1, BUF_PFN      ; 1212
          54          54 DD 001F7      ;      PUSHL     N      ; 1213
          7E          7E D4 001F9      ;      CLRL      -(SP)      ;
          8F          8F BB 001FB      ;      PUSHR     #^M<R6,R10>      ;
          04          04 FB 001FF      ;      CALLS     #4, MAP_SETUP      ;
;
; Now turn on the memory controllers Diagnostic Check Mode and
; Page
; Mode, and execute the DATI. After a nominal wait check that
; the
; UCE and ERR bits are set in the CSR.
;
          50          07 A3          0C 88 00204      ;      BISB2     #12, 7(MEM_CSR)      ; 1227
          60      0000G CF 0003F464 8F C9 00208      ;      BISL3     #259172, UNIBUS_SPACE, R0      ;
          59          01 A9 00212      ;      BISW3     #1, TEMP4, (R0)      ; 1229
          7E          0A 7D 00216      ;      MOVQ      #10, -(SP)      ; 1231
          00000000G 9F          02 FB 00219      ;      CALLS     #2, @DS$WAITUS      ;
          04          04 A3 D4 00220      ;      CLRL      4(MEM_CSR)      ; 1232
          63 E0000000 8F D0 00223      ;      MOVL      #-536870912, (MEM_CSR)      ; 1233
          55      0000GDF44 1FFFFFFE 8F CB 0022A      ;      BICL3     #536870910, @UBI_UNDER_TEST[N], TEMP1      ; 1234
          04          04 A3 D4 00235      ;      CLRL      4(MEM_CSR)      ; 1235
          57 A0000000 8F D0 00238      ;      MOVL      #-1610612736, TEMP2      ; 1236
          57          55 D1 0023F      ;      CMPL      TEMP1, TEMP2      ; 1237
          1C          13 00242      ;      BEQL      9$      ;
          7E          7C 00244      ;      CLRQ      -(SP)      ; 1242
          7E          7E D4 00246      ;      CLRL      -(SP)      ;
          55          55 DD 00248      ;      PUSHL     TEMP1      ;
          0090          8F BB 0024A      ;      PUSHR     #^M<R4,R7>      ;
          0000G CF 9F 0024E      ;      PUSHAB    PRINT_CSR_ERR      ;
          0000G CF 9F 00252      ;      PUSHAB    UBIMOD_CSR      ;
          7E          0000G CF 9A 00256      ;      MOVZBL    LUN, -(SP)      ;
          05          05 DD 0025B      ;      PUSHL     #5      ;
          6B          0A FB 0025D      ;      CALLS     #10, DS$ERRHARD      ;
;
; Check that the PE bit set in the UET.
;

```

```

50      0000G  CF 0003F464 8F C9 00260 9$: BISL3 #259172, UNIBUS_SPACE, R0 ; 1251
55      55 FFFFFFF7F 60 3C 0026A MOVZWL (R0), TEMP1 ;
55      8F CA 0026D BICL2 #-129, TEMP1 ; 1252
20      12 00274 BNEQ 10$ ; 1253
7E      7C 00276 CLRQ -(SP) ; 1259
55      DD 00278 PUSHL TEMP1 ;
7E      80 8F 9A 0027A MOVZBL #128, -(SP) ;
02      DD 0027E PUSHL #2 ;
0000G   CF 9F 00280 PUSHAB FMT_UET_STAT ;
0000G   CF 9F 00284 PUSHAB PRINT_GENERAL ;
0000G   CF 9F 00288 PUSHAB UETMOD ;
7E      0000G CF 9A 0028C MOVZBL LUN, -(SP) ;
06      DD 00291 PUSHL #6 ;
6B      0A FB 00293 CALLS #10, DS$ERRHARD ;

; Check that the UCE bit clears in the UBI CSR.
0000'   CF 0000GDF44 20000000 8F D0 00296 10$: MOVL #536870912, aUBI_UNDER_TEST[N] ; 1268
0000GDF44 1FFFFFFFE 8F CB 002A0 BICL3 #536870910, aUBI_UNDER_TEST[N], STATUS1 ; 1269
50      0000' CF D0 002AD MOVL STATUS1, R0 ;
1C      18 002B2 BGEQ 11$ ;
7E      7C 002B4 CLRQ -(SP) ;
7E      D4 002B6 CLRL -(SP) ;
50      DD 002B8 PUSHL R0 ;
7E      D4 002BA CLRL -(SP) ;
54      DD 002BC PUSHL N ;
0000G   CF 9F 002BE PUSHAB PRINT_CSR_ERR ;
0000G   CF 9F 002C2 PUSHAB UBIMOD_CSR ;
7E      0000G CF 9A 002C6 MOVZBL LUN, -(SP) ;
07      DD 002CB PUSHL #7 ;
6B      0A FB 002CD CALLS #10, DS$ERRHARD ;

; Check that the PE bit clears with a UET INIT.
50      0000G  CF 0003F465 8F C9 002D0 11$: BISL3 #259173, UNIBUS_SPACE, R0 ; 1276
60      80 8F 90 002DA MOVB #-128, (R0) ; 1277
50      0000G  CF 0003F464 8F C9 002DE BISL3 #259172, UNIBUS_SPACE, R0 ; 1278
55      55 FFFFFFF7F 60 3C 002E8 MOVZWL (R0), TEMP1 ;
55      8F CA 002EB BICL2 #-129, TEMP1 ;
1D      13 002F2 BEQL 12$ ; 1279
7E      7C 002F4 CLRQ -(SP) ; 1285
55      DD 002F6 PUSHL TEMP1 ;
7E      02 7D 002F8 MOVQ #2, -(SP) ;
0000G   CF 9F 002FB PUSHAB FMT_UET_STAT ;
0000G   CF 9F 002FF PUSHAB PRINT_GENERAL ;
0000G   CF 9F 00303 PUSHAB UETMOD ;
7E      0000G CF 9A 00307 MOVZBL LUN, -(SP) ;
08      DD 0030C PUSHL #8 ;
00000000G 6B 0A FB 0030E CALLS #10, DS$ERRHARD ;
9F      00 FB 00311 12$: CALLS #0, aDS$INLOOP ; 1290
03      50 E9 00318 BLBC R0, 14$ ;
F9      54 FE 7A 31 0031B 13$: BRW 8$ ;
03      F3 0031E 14$: AOBLEQ #3, N, 13$ ; 1147
02      DD 00322 PUSHL #2 ; 1292
1A      DD 00324 PUSHL #26 ;

```


00000000G 9F
00000000G 9F
50

02 FB 00326
00 FB 0032D
01 D0 00334
04 00337

CALLS #2, a#DS\$ENDSUB
CALLS #0, a#DS\$BREAK
MOVL #1, R0
RET

:
: 1294
: 1296
:

; Routine Size: 824 bytes, Routine Base: TEST_026 + 0000

; 1297 1
; 1298 1 END
; 1299 0 ELUDOM

PSECT SUMMARY									
Name	Bytes	Attributes							
\$OWNS	22	NOVEC, WRT, RD	NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)						
\$PLIT\$	132	NOVEC, NOWRT, RD	NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)						
DISPATCH	96	NOVEC, NOWRT, RD	NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)						
TEST_023	325	NOVEC, NOWRT, RD	EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)						
TEST_024	1128	NOVEC, NOWRT, RD	EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)						
TEST_025	316	NOVEC, NOWRT, RD	EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)						
TEST_026	824	NOVEC, NOWRT, RD	EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)						

Library Statistics						
File	-----		Symbols	-----		Processing Time
	Total	Loaded	Percent	Pages Mapped		
SYS\$COMMON:[SYSLIB]STARLET.L32;3	10093	1	0	598		00:00.8
SYS\$COMMON:[SYSLIB]DIAG.L32;265	784	19	2	85		00:00.3

COMMAND QUALIFIERS

BLISS/LIST ECCBA7.B32

Size: 2593 code + 250 data bytes

Run Time: 00:27.5

Elapsed Time: 00:35.9

Lines/CPU Min: 2835

Lexemes/CPU-Min: 31414

ZZ-ECCBA-1.4 TEST_026: CSR Status Bit Test
UBI_TESTS_23_26
01-04 TEST_026: CSR Status Bit Test

; Memory Used: 359 pages
; Compilation Complete

G 2

6-Sep-1985

Fiche 3 Frame G2

Sequence 431

6-Sep-1985 17:24:39

VAX-11 Bliss-32 V4.1-746

Page 48

```

; 0001 0  MODULE UBI_TESTS_27_32 (      ! UBI diagnostic program tests 27 to 32
; 0002 0      IDENT = '01-04'
; 0003 0      ) =
; 0004 1  BEGIN
; 0005 1
; 0006 1  !
; 0007 1  ! COPYRIGHT (C) 1980,1981
; 0008 1  ! DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
; 0009 1  !
; 0010 1  ! THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
; 0011 1  ! COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
; 0012 1  ! ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
; 0013 1  ! MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
; 0014 1  ! EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
; 0015 1  ! TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
; 0016 1  ! REMAIN IN DEC.
; 0017 1  !
; 0018 1  ! THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
; 0019 1  ! AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
; 0020 1  ! CORPORATION.
; 0021 1  !
; 0022 1  ! DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
; 0023 1  ! SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
; 0024 1  !
; 0025 1  !
; 0026 1  ! **
; 0027 1  ! FACILITY:      VAX-11 Diagnostic Library
; 0028 1  !
; 0029 1  ! ABSTRACT:      This module contains tests 23 to 29 of the COMET
; 0030 1  !                  Unibus Interface Diagnostic Program.
; 0031 1  !
; 0032 1  ! ENVIRONMENT:    VAX-11 Diagnostic Supervisor
; 0033 1  !
; 0034 1  ! AUTHOR:          Rand Gray, CREATION DATE: 11-Dec-78
; 0035 1  !
; 0036 1  ! MODIFIED BY:     Don Monroe
; 0037 1  !
; 0038 1  !             0-7      May 1979
; 0039 1  !                  Changed the format of the UBI MAP registers.
; 0040 1  !
; 0041 1  !             0-8      July 1979
; 0042 1  !                  Modified to support the real UET module.
; 0043 1  !
; 0044 1  !                  Fixed a BUG in XFER_SETUP routine. Loop that setup the maps
; 0045 1  !                  would not work if MAP_NUM was not zero.
; 0046 1  !
; 0047 1  !             0-9      July 1979
; 0048 1  !                  Changed referances to 'Byte Offset' to 'Byte Number'
; 0049 1  !
; 0050 1  !
; 0051 1  !             0-10     October 1979
; 0052 1  !                  Added a return value to the MAP_BUFFERS routine to support
; 0053 1  !                  the changes to Module 7 Revision 10.
; 0054 1  !                  Added byte-offset parameter to MAP_BUFFERS routine so routine
; 0055 1  !                  returns with one additional page per buffer.

```



```
; 0056 1 !
; 0057 1 !
; 0058 1 ! 0-11 October 22,1979
; 0059 1 ! Added a routine to print MAP errors. Called via PRINT LINK
; 0060 1 ! in error hard calls.
; 0061 1 !
; 0062 1 ! 0-12 December 7,1979
; 0063 1 ! Fixed bug in PROBE_ADDRESS routine in the ADAWI builtin.
; 0064 1 !
; 0065 1 ! 0-13 April 7,1980
; 0066 1 ! Changed address of unibus space so second UBI will work.
; 0067 1 !
; 0068 1 ! 0-14 April 8,1980
; 0069 1 ! Added a print general routine and converted PRINTX after
; 0070 1 ! error calls to print links.
; 0071 1 ! 1-00 June 16,1980
; 0072 1 ! Initial Release
; 0073 1 ! 1-01 July 31,1980
; 0074 1 ! Fixed bug in MAP_BUFFERS routine that would not detect
; 0075 1 ! insufficient memory for the requested number of buffers.
; 0076 1 ! 1-02 December 29,1980
; 0077 1 ! Fixed bug in MAP_BUFFERS routine. Supervisor allocates memory
; 0078 1 ! even if not enough for requested buffer size.
; 0079 1 ! 1-03 May 6, 1981
; 0080 1 ! Fixed bug in map_buffers routine.
; 0081 1 !
; 0082 1 ! Kevin LeMieux
; 0083 1 !
; 0084 1 ! 1-04 February,1985
; 0085 1 ! Fixed bug in INIT code. If two DW's were specified; after the
; 0086 1 ! first pass one of them one of them was no longer tested.
; 0087 1 ! Fixed minor bugs in TEST 29.
; 0087 1 !--
```

```

; 0088 1 !
; 0089 1 ! TABLE OF CONTENTS:
; 0090 1 !
; 0091 1 ! UET CSR1 Interrupt Test
; 0092 1 ! UET CSR2 Interrupt Test
; 0093 1 ! UET CSR1/CSR2 ACLOW Test
; 0094 1 ! Map Invalid Test
; 0095 1 ! UET PB Bit Test
; 0096 1 ! UBE Block Transfer Test
; 0097 1 !
; 0098 1 !
; 0099 1 !
; 0100 1 ! INCLUDE FILES:
; 0101 1 !
; 0102 1 !
; 0103 1 LIBRARY 'SYS$LIBRARY:STARLET';
; 0104 1 LIBRARY 'SYS$LIBRARY:DIAG';
; 0105 1 !
; 0106 1 !
; 0107 1 ! MACROS:
; 0108 1 !
; 0109 1 !
; 0110 1 !
; 0111 1 ! EQUATED SYMBOLS:
; 0112 1 !
; 0113 1 !
; 0114 1 LITERAL
; 0115 1 IO_RESET = %X'37';
; 0116 1
; 0117 1 LITERAL
; 0118 1 ALL_ONES = %X'FFFFFFFF',
; 0119 1 ALL_ZEROS = 0,
; 0120 1
; 0121 1 BR4 = %X'100',
; 0122 1 BR5 = %X'200',
; 0123 1 BR6 = %X'400',
; 0124 1 BR7 = %X'800',
; 0125 1
; 0126 1 CLEAR_MEM_ERR = %X'E0000000', ! A8
; 0127 1 CSR_MASK = %X'E0000001',
; 0128 1
; 0129 1 DAT1 = 1,
; 0130 1 DATIP = 3,
; 0131 1 DATO = 5,
; 0132 1 DATOB = 7,
; 0133 1
; 0134 1 DIAG_CHK_MODE = %X'C0000000',
; 0135 1 DISABLE_ECC = %X'20000000',
; 0136 1 ERR = BIT31,
; 0137 1 MAP_MASK = %X'82607FFF', ! M7
; 0138 1 MEMORY_CONT = %X'F20000',
; 0139 1 NON_XIST_NEXUS = %X'F40000',
; 0140 1 NXM = BIT30,
; 0141 1 PAGE_MODE = %X'80000000', ! A8
; 0142 1 PURGE = BIT0,

```

```

; 0143 1      UCE = BIT29;
; 0144 1      !
; 0145 1      ! OWN STORAGE:
; 0146 1      !
; 0147 1      ! OWN
; 0148 1      BYTE_PATTERN: VECTOR[4,BYTE],      ! Data patterns loaded at run time
; 0149 1      STATUS0: SIGNED WORD,              ! Used by the UET status macro
; 0150 1      STATUS1,                          ! Used by the UBI status macro
; 0151 1      WORD_PATTERN: VECTOR[5,WORD];      ! Data patterns loaded at run time
; 0152 1
; 0153 1      !
; 0154 1      ! EXTERNAL REFERENCES:
; 0155 1      !
; 0156 1      EXTERNAL ROUTINE
; 0157 1      BUFFER_SETUP,
; 0158 1      DECODER,
; 0159 1      ENCODER,
; 0160 1      MAP_BUFFERS,
; 0161 1      MAP_SETUP,
; 0162 1      PRINT_GENERAL:NOVALUE,
; 0163 1      PROBE_ADDRESS,
; 0164 1      UNIBUS_SIZER,
; 0165 1      XFER_SETUP;
; 0166 1
; 0167 1      EXTERNAL
; 0168 1      CHAN_STAT,
; 0169 1      CODEWORDS: VECTOR[9,WORD],
; 0170 1      DECODED_WORDS: VECTOR[9],
; 0171 1      EVENT_FLAG: BITVECTOR[4],
; 0172 1      EXTERNAL_FLAG: BYTE,
; 0173 1      FMT_BDP_DATI,
; 0174 1      FMT_BDP_DATO,
; 0175 1      FMT_BDP_DATO,
; 0176 1      FMT_BYTE_OFF,
; 0177 1      FMT_CMI_UB,
; 0178 1      FMT_CPU_RW,
; 0179 1      FMT_DDP_DATI,
; 0180 1      FMT_DDP_DATO,
; 0181 1      FMT_DDP_DATO,
; 0182 1      FMT_DP_SEL,
; 0183 1      FMT_EXT_PROC,
; 0184 1      FMT_INIT_ACLO,
; 0185 1      FMT_MAP_ADDR,
; 0186 1      FMT_MAP_CS,
; 0187 1      FMT_MAP_FAIL,
; 0188 1      FMT_MAP_INV,
; 0189 1      FMT_MCHK,
; 0190 1      FMT_NO_ACINT,
; 0191 1      FMT_NO_ACINT1,
; 0192 1      FMT_NO_INT,
; 0193 1      FMT_NO_MCHK,
; 0194 1      FMT_NO_UB,
; 0195 1      FMT-UA1,
; 0196 1      FMT_UB_CMI,
; 0197 1      FMT_UB_TO,

```

! A8

! A8

! A8


```

; 0198 1 FMT_UBE_DUMP,
; 0199 1 FMT_UBI_DUMP,
; 0200 1 FMT_UET_ACLO1,
; 0201 1 FMT_UET_INTD,
; 0202 1 FMT_UET_INTD2,
; 0203 1 FMT_UET_PB, ! A8
; 0204 1 FMT_UET_STAT,
; 0205 1 FMT_UNX_ACINT,
; 0206 1 FMT_UNX_ACINT1,
; 0207 1 FMT_UNX_ACINT2,
; 0208 1 FMT_UNX_INT1,
; 0209 1 FMT_VECTOR,
; 0210 1 FREE_MAP: VECTOR[4,WORD],
; 0211 1 HANDLER,
; 0212 1 INT_VECTOR,
; 0213 1 INTERRUPT_EXP: BYTE,
; 0214 1 INTERRUPT_OCC: BYTE,
; 0215 1 LUN: BYTE,
; 0216 1 NOISY_WORDS: VECTOR[9,WORD],
; 0217 1 NUM_UBE,
; 0218 1 PRINT_CSR_ERR,
; 0219 1 PRINT_DCMF_ERR,
; 0220 1 PRINT_MAP_ERR, ! A13
; 0221 1 UBE_BUF_ADDR: VECTOR[],
; 0222 1 UBE_INT_OCC: VECTOR[,BYTE], ! A8
; 0223 1 UBE_SERVICE: VECTOR[], ! A8
; 0224 1 UBE_VECTOR: VECTOR[,WORD], ! A8
; 0225 1 UBIMOD,
; 0226 1 UBIMOD_BDP,
; 0227 1 UBIMOD_BYTE_OFF,
; 0228 1 UBIMOD_CMI,
; 0229 1 UBIMOD_CPU_RW,
; 0230 1 UBIMOD_CSR,
; 0231 1 UBIMOD_DDP,
; 0232 1 UBIMOD_DP_SEL,
; 0233 1 UBIMOD_MAP,
; 0234 1 UBIMOD_MAP_CS,
; 0235 1 UBIMOD_MAP_ADDR,
; 0236 1 UBIMOD_MAP_CTRL,
; 0237 1 UBIMOD_UA1,
; 0238 1 UBIMOD_UB_CMI,
; 0239 1 UBI_VECTOR:WORD,
; 0240 1 UBI_LINK: REF $DS_HPO_DECL(),
; 0241 1 UETMOD, ! A8
; 0242 1 SCRATCH: VECTOR[128],
; 0243 1 UBE_BASE_ADDR: VECTOR[4],
; 0244 1 UBE_BUF_SIZE,
; 0245 1 UBE_DRIVE: VECTOR[4,BYTE],
; 0246 1 UBE_PAGE_CNT,
; 0247 1 UBE_INIT_DATA: VECTOR[4,WORD], ! A10
; 0248 1 UBE_MODE: VECTOR[4,BYTE], ! A10
; 0249 1 UBE_MAP_NO: VECTOR[4,WORD], ! A10
; 0250 1 UBE_DP_NO: VECTOR[4,BYTE], ! A10
; 0251 1 UBE_EXP_DATA: VECTOR[4,WORD],
; 0252 1 UBE_XFER_ERR: BITVECTOR[4],

```

```

; 0253 1 UBE_STAT_ERR: BITVECTOR[4],
; 0254 1 UBE0_ISR,
; 0255 1 UBI_UNDER_TEST,
; 0256 1 UNIBUS_SPACE,
; 0257 1 UET_ISR,
; 0258 1 UET_ISR_2,
; 0259 1 VALID_MAPS: BITVECTOR[512];
; 0260 1
; 0261 1 MACRO
; 0262 1 !++
; 0263 1 ! This macro defines the address of the control and status register
; 0264 1 ! for the given buffered data path number.
; 0265 1 !--
; 0266 1 UBI_CSR(BDP) = (.UBI_UNDER_TEST + BDP*4)x,
; 0267 1
; 0268 1 !++
; 0269 1 ! This macro defines the address of a UBI map for the given map
; 0270 1 ! number.
; 0271 1 !--
; 0272 1 UBI_MAP(NUM) = (.UBI_UNDER_TEST + xX'800' + NUM*4)x;
; 0273 1
; 0274 1 !++
; 0275 1 ! These macros temporarily define the addresses of the (simulated) UET
; 0276 1 ! registers. They will be replaced for the real UET.
; 0277 1 !--
; 0278 1 MACRO ! A8
; 0279 1 UET_ADDR_REG = (.UNIBUS_SPACE OR xX'3F460')<0,16>x, ! A8 M15
; 0280 1 UET_DATA_REG = (.UNIBUS_SPACE OR xX'3F462')<0,16>x, ! A8 M15
; 0281 1 UET_DATA_REGB = (.UNIBUS_SPACE OR xX'3F462')<0,8>x, ! A8 M15
; 0282 1 UET_CTRL_REG = (.UNIBUS_SPACE OR xX'3F464')<0,16>x, ! A8 M15
; 0283 1 UET_CTRL_REGB = (.UNIBUS_SPACE OR xX'3F465')<0,8>x, ! A8 M15
; 0284 1 UET_CTRL_REG2 = (.UNIBUS_SPACE OR xX'3F466')<0,16>x,
; 0285 1
; 0286 1 !++ ! A8
; 0287 1 ! This macro is used for checking the status of the UET. ! A8
; 0288 1 !-- ! A8
; M 0289 1 UET_STATUS(CALLOUT) = ! A8
; M 0290 1 STATUS0 = .UET_CTRL_REG; ! Read the UET control register ! A8
; M 0291 1 STATUS0 = .STATUS0 AND xX'E1'; ! A8
; M 0292 1 IF .STATUS0 NEQ 0 ! A8
; M 0293 1 THEN ! A8
; M 0294 1 BEGIN ! A8
; M 0295 1 $DS_ERRHARD(UNIT=.LUN,MSGADR=CALLOUT,
; M 0296 1 PRLINK = PRINT_GENERAL,
; M 0297 1 P1 = FMT_UET_STAT,P2 = 2,
; M 0298 1 P3 = 0,P4 = .STATUS0); ! A8
; 0299 1 END;x; ! A8
; 0300 1 MACRO ! A8
; 0301 1 !++
; 0302 1 ! This macro is used for checking the status of the UBI.
; 0303 1 !--
; M 0304 1 UBI_STATUS(BDP,CALLOUT) =
; M 0305 1 STATUS1 = .UBI_CSR(BDP) AND CSR_MASK;
; M 0306 1 IF .STATUS1 LSS 0
; M 0307 1 THEN

```

ZZ-ECCBA-1.4 01-04
UBI_TESTS_27_32
01-04

N 2
6-Sep-1985 Fiche 3 Frame N2 Sequence 438
6-Sep-1985 17:25:18 VAX-11 Bliss-32 V4.1-746
6-Sep-1985 17:15:07 [LEMIEUX.ECCBA.RELEASE]ECCBA8.B32;19 Page 7
(2)

```
: M 0308 1          $DS_ERRHARD(UNIT=.LUN,  
: M 0309 1          MSGADR=CALLOUT,  
: M 0310 1          PRLINK=PRINT_CSR_ERR,  
: 0311 1          P1=BDP,P2=0,P3=.STATUS1);x;  
: 0312 1  
: 0313 1  
: 0314 1          ! Required Diagnostic Supervisor macros.  
: 0315 1          !  
: L 0316 1          $DS_BGNMOD(ENV=0,TEST=27);  
: xPRINT:          Bliss-32 Diagnostic Macro Library version 7.0 "DIAG.L32 (57)"  
: 0317 1          $DS_DSADEF;  
: 0318 1          $DS_SECDEF(ALL,UBE);  
: 0319 1  
: 0320 1          BUILTIN  
: 0321 1          ROT,  
: 0322 1          MTPR;
```


TEST_027: UET CSR1 Interrupt Test

```

: 0323 2 $DS_BGNTEST (SECTION=(DEFAULT,ALL),TEXT='UET CSR1 Interrupt Test');
: 0324 2
: 0325 2 !**
: 0326 2 ! TEST DESCRIPTION:
: 0327 2 !
: 0328 2 ! This tests the ability of the UBI to handle interrupts at BR4-BR7,
: 0329 2 ! using the UET to initiate interrupts at each BR
: 0330 2 ! level. The UET has a programmable vector, which for this test is
: 0331 2 ! arbitrarily chosen as 340 (hex), which is just below the vectors for
: 0332 2 ! the UBEs.
: 0333 2 !
: 0334 2 ! If the interrupt occurs, the interrupt service routine clears the
: 0335 2 ! BR request bit in the UET Control Register.
: 0336 2 !
: 0337 2 ! Subtest 5 then floats a one and a zero thru the vector and forces
: 0338 2 ! an interrupt on level 4. All vectors for the Unibus are captured
: 0339 2 ! so that failures in the interrupt vector will not be reported by
: 0340 2 ! the Supervisor.
: 0341 2 !
: 0342 2 ! ASSUMPTIONS:
: 0343 2 !
: 0344 2 ! Test 1-Test 15
: 0345 2 ! --
: 0346 2
: 0347 2 REGISTER
: 0348 2 IPL,
: 0349 2 TEMP: WORD,
: 0350 2 I;
: 0351 2
: 0352 3 $DS_BGNSUB; ! Subtest 1: BR7
: 0353 3
: 0354 3 UET_DATA_REG = %X'140';
: 0355 3 $DS_CHANNEL(UNIT=.LUN,FUNC=CHC$_ENINT,VECADR=UET_ISR,STSADR=CHAN_STAT);
: 0356 3
: 0357 3 CODECOMMENT
: 0358 3 ''
: 0359 3 'Set the IPL to 1F (hex), set the interrupt expected flag, clear the',
: 0360 3 'interrupt occurred flag, and initiate an interrupt at BR7. Lower the IPL',
: 0361 3 'from 1F to 17 by steps of one, checking each time that the interrupt has',
: 0362 3 'not occurred.',
: 0363 3 '';
: 0364 4 BEGIN
: 0365 4
: 0366 4 DECR IPL FROM %X'1F' TO %X'17' DO
: 0367 5 BEGIN
: 0368 5 $DS_SETIPL(LEVEL=%X'1F');
: 0369 5 DO ! Scope loop starts here
: 0370 6 BEGIN
: 0371 6 UET_CTRL_REG = %X'8000';
: 0372 6 $DS_SETIPL(LEVEL=.IPL);
: 0373 6 INTERRUPT_EXP = 1;
: 0374 6 INTERRUPT_OCC = 0;
: 0375 6 UET_CTRL_REG = BR7;
: 0376 6
: 0377 6 $DS_WAITUS(TIME=20);

```

! A8

```

: 0378 6      IF .INTERRUPT_OCC
: 0379 6      THEN
: 0380 6          CODECOMMENT
: 0381 6          ''
: 0382 6          'Interrupt occurred at IPL greater than or equal to 17',
: 0383 6          ''
: 0384 7          BEGIN
: P 0385 7          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0386 7              PRLINK = PRINT_GENERAL,
: P 0387 7              P1 = FMT_UNX_INT1,P2=2,
: L 0388 7              P3 = 7,P4 = .IPL);
: XPRINT:      Test 27, Subtest 1, Error 1
: 0389 7          END
: 0390 6      ELSE
: 0391 6          CODECOMMENT
: 0392 6          ''
: 0393 6          'If this is unit 1 then check that interrupt done bit is not set',
: 0394 6          ''
: 0395 7          BEGIN
: 0396 7          IF .UBI_LINK[HP$B_DRIVE] EQL 1
: 0397 7          THEN
: 0398 8              BEGIN
: 0399 8              TEMP = .UET_CTRL_REG;
: 0400 8              IF (.TEMP AND %X'4000') NEQ 0
: 0401 8              THEN
: P 0402 8                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0403 8                      PRLINK = PRINT_GENERAL,
: P 0404 8                      P1 = FMT_UET_INTD,P2 = 2,
: L 0405 8                      P3 = 0,P4 = .TEMP AND %X'4000');
: XPRINT:      Test 27, Subtest 1, Error 2
: 0406 7          END;
: 0407 6          END;
: 0408 6
: 0409 6          END
: 0410 5          WHILE $DS_INLOOP;          ! Scope loop ends here
: 0411 4          END;
: 0412 3          END;
: 0413 3
: 0414 3      CODECOMMENT
: 0415 3      ''
: 0416 3      'Now lower the IPL to 16 and after a nominal wait check to see if the',
: 0417 3      'interrupt has occurred.',
: 0418 3      ''
: 0419 4          BEGIN
: 0420 4
: 0421 4          DO          ! Scope loop begins here
: 0422 5              BEGIN
: 0423 5              UET_CTRL_REG = %X'8000';
: 0424 5              $DS_SETIPL(LEVEL=%X'16');
: 0425 5              INTERRUPT_EXP = 1;
: 0426 5              INTERRUPT_OCC = 0;
: 0427 5              UET_CTRL_REG = BR7;          ! A10
: 0428 5              $DS_WAITUS(TIME=20);
: 0429 5              IF NOT .INTERRUPT_OCC
: 0430 5              THEN

```

```

: 0431 6      BEGIN
: P 0432 6      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0433 6          PRLINK = PRINT_GENERAL,
: P 0434 6          P1 = FMT_NO_INT,P2 = 2,
: L 0435 6          P3 = 7,P4 = 'X'16');
: xPRINT:      Test 27, Subtest 1, Error 3
: 0436 6      END
: 0437 5      ELSE
: 0438 5      CODECOMMENT
: 0439 5      ..
: 0440 5      'If this is unibus number 1 check that the interrupt done',
: 0441 5      'bit in CSR1 is set. Then check that it does not clear when',
: 0442 5      'it is written to a zero and that it does clear when written',
: 0443 5      'to a one.',
: 0444 5      ..
: 0445 6      BEGIN
: 0446 6      IF .UBI_LINK[HP$B_DRIVE] EQL 1
: 0447 6      THEN
: 0448 7      BEGIN
: 0449 7      TEMP = .UET_CTRL_REG;
: 0450 7      IF (.TEMP AND 'X'4000') NEQ 'X'4000'
: 0451 7      THEN
: 0452 7      CODECOMMENT
: 0453 7      ..
: 0454 7      'Interrupt done bit in CSR1 did not set.',
: 0455 7      ..
: 0456 8      BEGIN
: P 0457 8      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0458 8          PRLINK = PRINT_GENERAL,
: P 0459 8          P1 = FMT_UET_INTD,P2 = 2,
: L 0460 8          P3 = 'X'4000',P4 = .TEMP AND 'X'4000');
: xPRINT:      Test 27, Subtest 1, Error 4
: 0461 8      END
: 0462 7      ELSE
: 0463 7      CODECOMMENT
: 0464 7      ..
: 0465 7      'Check that Interrupt Done bit clears properly. First',
: 0466 7      'try and clear it by writting a zero and check that',
: 0467 7      'that it does not clear.',
: 0468 7      ..
: 0469 8      BEGIN
: 0470 8      UET_CTRL_REG = 0;
: 0471 8      TEMP = .UET_CTRL_REG;
: 0472 8      IF (.TEMP AND 'X'4000') NEQ 'X'4000'
: 0473 8      THEN
: 0474 8      CODECOMMENT
: 0475 8      ..
: 0476 8      'Interrupt done bit cleared when writting a zero.',
: 0477 8      ..
: 0478 9      BEGIN
: P 0479 9      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0480 9          PRLINK = PRINT_GENERAL,
: P 0481 9          P1 = FMT_UET_INTD,P2 = 2,
: L 0482 9          P3 = 'X'4000',P4=.TEMP AND 'X'4000');
: xPRINT:      Test 27, Subtest 1, Error 5

```



```

; 0483 9
; 0484 8
; 0485 8
; 0486 8
; 0487 8
; 0488 8
; 0489 9
; 0490 9
; 0491 9
; 0492 9
; 0493 9
; P 0494 9
; P 0495 9
; P 0496 9
; L 0497 9
; xPRINT:
; 0498 8
; 0499 7
; 0500 6
; 0501 5
; 0502 5
; 0503 4
; 0504 4
; 0505 3
; 0506 3
; 0507 2
; 0508 2
; 0509 3
; 0510 3
; 0511 3
; 0512 3
; 0513 3
; 0514 3
; 0515 3
; 0516 3
; 0517 3
; 0518 4
; 0519 4
; 0520 4
; 0521 5
; 0522 5
; 0523 5
; 0524 6
; 0525 6
; 0526 6
; 0527 6
; 0528 6
; 0529 6
; 0530 6
; 0531 6
; 0532 6
; 0533 6
; P 0534 6
; P 0535 6
; P 0536 6

END
ELSE
CODECOMMENT
,,
' Check that it clears when a 1 is written to it.',
,,
BEGIN
UET_CTRL_REG = %X'4000';
TEMP = .UET_CTRL_REG;
IF (.TEMP AND %X'4000') NEQ 0
THEN
$DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
PRLINK=PRINT_GENERAL,
P1=FMT_UET_INTD,P2=2,
P3 = 0,P4=.TEMP AND %X'4000');

Test 27, Subtest 1, Error 6
END;
END;
END;
END;
WHILE $DS_INLOOP; ! Scope loop ends here
END;
$DS_ENDSUB; ! End of Subtest 1
$DS_BGNSUB; ! Subtest 2: BR6
CODECOMMENT
,,
'Set the IPL to 1F (hex), set the interrupt expected flag, clear the',
'interrupt occurred flag, and initiate an interrupt at BR6. Lower the IPL',
'from 1F to 16 by steps of one, checking each time that the interrupt has',
'not occurred.',
,,
BEGIN
DECR IPL FROM %X'1F' TO %X'16' DO
BEGIN
$DS_SETIPL(LEVEL=%X'1F');
DO ! Scope loop starts here
BEGIN
UET_CTRL_REG = %X'8000';
$DS_SETIPL(LEVEL=.IPL);
INTERRUPT_EXP = 1;
INTERRUPT_OCC = 0;
UET_CTRL_REG = BR6;

$DS_WAITUS(TIME=20); ! A8
IF .INTERRUPT_OCC
THEN
$DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
PRLINK = PRINT_GENERAL,
P1 = FMT_UNX_INT1,P2 = 2,

```

```
; L 0537 6
; xPRINT:
; 0538 6
; 0539 6
; 0540 5      END
; 0541 4      WHILE $DS_INLOOP;          ! Scope loop ends here
; 0542 3      END;
; 0543 3      END;
; 0544 3      CODECOMMENT
; 0545 3      ''
; 0546 3      'Now lower the IPL to 15 and after a nominal wait check to see if the',
; 0547 3      'interrupt has occurred.',
; 0548 3      ''
; 0549 4      BEGIN
; 0550 4
; 0551 4      DO          ! Scope loop begins here
; 0552 5      BEGIN
; 0553 5      UET_CTRL_REG = xX'8000';
; 0554 5      INTERRUPT_EXP = 1;
; 0555 5      INTERRUPT_OCC = 0;
; 0556 5      $DS_SETIPL(LEVEL=xX'15');
; 0557 5      UET_CTRL_REG = BR6;          ! A10
; 0558 5      $DS_WAITUS(TIME=20);
; 0559 5      IF NOT .INTERRUPT_OCC
; 0560 5      THEN
; P 0561 5      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; P 0562 5      PRLINK = PRINT_GENERAL,
; P 0563 5      P1 = FMT_NO_INT,P2 = 2,
; L 0564 5      P3 = 6,P4 = 15);
; xPRINT:
; 0565 5      Test 27, Subtest 2, Error 8
; 0566 4      END
; 0567 4      WHILE $DS_INLOOP;          ! Scope loop ends here
; 0568 3      END;
; 0569 3
; 0570 2      $DS_ENDSUB;          ! End of Subtest 2
; 0571 2
; 0572 3      $DS_BGNSUB;          ! Subtest 3: BR5
; 0573 3
; 0574 3      CODECOMMENT
; 0575 3      ''
; 0576 3      'Set the IPL to 1F (hex), set the interrupt expected flag, clear the',
; 0577 3      'interrupt occurred flag, and initiate an interrupt at BR5. Lower the IPL',
; 0578 3      'from 1F to 15 by steps of one, checking each time that the interrupt has',
; 0579 3      'not occurred.',
; 0580 3      ''
; 0581 4      BEGIN
; 0582 4
; 0583 4      DECR IPL FROM xX'1F' TO xX'15' DO
; 0584 5      BEGIN
; 0585 5      $DS_SETIPL(LEVEL=xX'1F');
; 0586 5      DO          ! Scope loop starts here
; 0587 6      BEGIN
; 0588 6      UET_CTRL_REG = xX'8000';
; 0589 6      $DS_SETIPL(LEVEL=.IPL);
```

```

: 0590 6      INTERRUPT_EXP = 1;
: 0591 6      INTERRUPT_OCC = 0;
: 0592 6      UET_CTRL_REG = BR5;
: 0593 6
: 0594 6      $DS_WAITUS(TIME=20);
: 0595 6      IF .INTERRUPT_OCC
: 0596 6      THEN
: P 0597 6          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0598 6          PRLINK = PRINT_GENERAL,
: P 0599 6          P1 = FMT_UNX_INT1,P2 = 2,
: L 0600 6          P3 = 5,P4 = .IPL);
: xPRINT:      Test 27, Subtest 3, Error 9
: 0601 6
: 0602 6      END
: 0603 5      WHILE $DS_INLOOP;          ! Scope loop ends here
: 0604 4      END;
: 0605 3      END;
: 0606 3
: 0607 3      CODECOMMENT
: 0608 3      ``
: 0609 3      'Now lower the IPL to 14 and after a nominal wait check to see if the',
: 0610 3      'interrupt has occurred.',
: 0611 3      ``
: 0612 4      BEGIN
: 0613 4
: 0614 4      DO          ! Scope loop begins here
: 0615 5      BEGIN
: 0616 5      UET_CTRL_REG = xX'8000';
: 0617 5      INTERRUPT_EXP = 1;
: 0618 5      INTERRUPT_OCC = 0;
: 0619 5      $DS_SETIPL(LEVEL=xX'14');
: 0620 5      UET_CTRL_REG = BR5;          ! A10
: 0621 5      $DS_WAITUS(TIME=20);
: 0622 5      IF NOT .INTERRUPT_OCC
: 0623 5      THEN
: P 0624 5          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0625 5          PRLINK = PRINT_GENERAL,
: P 0626 5          P1 = FMT_NO_INT,P2 = 2,
: L 0627 5          P3 = 5,P4 = 14);
: xPRINT:      Test 27, Subtest 3, Error 10
: 0628 5      END
: 0629 4      WHILE $DS_INLOOP;          ! Scope loop ends here
: 0630 4
: 0631 3      END;
: 0632 3
: 0633 2      $DS_ENDSUB;          ! End of Subtest 3
: 0634 2
: 0635 3      $DS_BGNSUB;          ! Subtest 4: BR4
: 0636 3
: 0637 3      CODECOMMENT
: 0638 3      ``
: 0639 3      'Set the IPL to 1F (hex), set the interrupt expected flag, clear the',
: 0640 3      'interrupt occurred flag, and initiate an interrupt at BR4. Lower the IPL',
: 0641 3      'from 1F to 14 by steps of one, checking each time that the interrupt has',
: 0642 3      'not occurred.',

```



```

: 0643 3  '':
: 0644 4  BEGIN
: 0645 4
: 0646 4  DECR IPL FROM %X'1F' TO %X'14' DO
: 0647 5  BEGIN
: 0648 5  $DS_SETIPL(LEVEL=%X'1F');
: 0649 5  DO                                     ! Scope loop starts here
: 0650 6  BEGIN
: 0651 6  UET_CTRL_REG = %X'8000';
: 0652 6  $DS_SETIPL(LEVEL=.IPL);
: 0653 6  INTERRUPT_EXP = 1;
: 0654 6  INTERRUPT_OCC = 0;
: 0655 6  UET_CTRL_REG = BR4;
: 0656 6
: 0657 6  $DS_WAITUS(TIME=20);
: 0658 6  IF .INTERRUPT_OCC
: 0659 6  THEN
: P 0660 6  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0661 6  PRLINK = PRINT_GENERAL,
: P 0662 6  P1 = FMT_UNX_INT1,P2 = 2,
: L 0663 6  P3 = 4,P4 = .IPL);
: %PRINT: Test 27, Subtest 4, Error 11
: 0664 6
: 0665 6  END
: 0666 5  WHILE $DS_INLOOP;                       ! Scope loop ends here
: 0667 4  END;
: 0668 3  END;
: 0669 3
: 0670 3  CODECOMMENT
: 0671 3  'Now lower the IPL to 13 and after a nominal wait check to see if the',
: 0672 3  'interrupt has occurred.',
: 0673 3  '':
: 0674 3  BEGIN
: 0675 4
: 0676 4  DO                                     ! Scope loop begins here
: 0677 4  BEGIN
: 0678 5  UET_CTRL_REG = %X'8000';
: 0679 5  INTERRUPT_EXP = 1;
: 0680 5  INTERRUPT_OCC = 0;
: 0681 5  $DS_SETIPL(LEVEL=%X'13');
: 0682 5  UET_CTRL_REG = BR4;
: 0683 5  $DS_WAITUS(TIME=20);
: 0684 5  IF NOT .INTERRUPT_OCC
: 0685 5  THEN
: 0686 5  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0687 5  PRLINK = PRINT_GENERAL,
: P 0688 5  P1 = FMT_NO_INT,P2 = 2,
: P 0689 5  P3 = 4,P4 = 13);
: L 0690 5
: %PRINT: Test 27, Subtest 4, Error 12
: 0691 5  END
: 0692 4  WHILE $DS_INLOOP;                       ! Scope loop ends here
: 0693 4  END;
: 0694 3
: 0695 3

```

! A8

! Scope loop begins here

! A10

! Scope loop ends here

```

: 0696 2 $DS_ENDSUB; ! End of Subtest 4
: 0697 2
: 0698 2 !
: 0699 2 ! The following Subtest was added in Revision 1-8.
: 0700 2 !
: 0701 3 $DS_BGNSUB;
: 0702 3
: 0703 3 CODECOMMENT
: 0704 3 ``
: 0705 3 'If there are UBEs selected, release their vectors.',
: 0706 3 ``
: 0707 4 BEGIN
: 0708 4 IF .NUM_UBE NEQ 0
: 0709 4 THEN
: 0710 4 INCR I FROM 0 TO .NUM_UBE - 1 DO
: 0711 4 $DS_CLRVEC(VECTOR=.UBE_VECTOR[I]);
: 0712 3 END;
: 0713 3
: 0714 3 CODECOMMENT
: 0715 3 ``
: 0716 3 'Set the IPL to 13, then float a ONE thru the Interrupt Vector (bits <8:2>)',
: 0717 3 'of the UET, forcing an interrupt on level 4. After each Interrupt, check',
: 0718 3 'the VECTOR that the interrupt occurred at.',
: 0719 3 ``
: 0720 4 BEGIN
: 0721 4 LOCAL
: 0722 4 EXP_VECTOR;
: 0723 4
: 0724 4 $DS_SETIPL(LEVEL=XX'13');
: 0725 4
: 0726 4 INCR I FROM 2 TO 8 DO
: 0727 5 BEGIN
: 0728 5 EXP_VECTOR = (1 ^ .I);
: 0729 5 UET_DATA_REG = .EXP_VECTOR;
: 0730 5 DO ! Scope loop begins here
: 0731 6 BEGIN
: 0732 6 UET_CTRL_REG = XX'8000';
: 0733 6 $DS_WAITUS(TIME=20);
: 0734 6 INTERRUPT_EXP = 1;
: 0735 6 UET_CTRL_REG = BR4; ! Force the Interrupt
: 0736 6 $DS_WAITUS(TIME=20);
: 0737 6 IF .INT_VECTOR NEQ .EXP_VECTOR
: 0738 6 THEN
: 0739 7 BEGIN
: 0740 7 $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD,
: 0741 7 PRLINK = PRINT_GENERAL,
: 0742 7 P1 = FMT_VECTOR,P2 = 2,
: 0743 7 P3 = .EXP_VECTOR,P4 = .INT_VECTOR);
: 0744 6 END;
: 0745 6 END
: 0746 5 WHILE $DS_INLOOP; ! Scope loop ends here
: 0747 4 END;
: 0748 3 END;
: 0749 3

```

Test 27, Subtest 5, Error 13

```

: 0750 3 CODECOMMENT
: 0751 3 ''
: 0752 3 'Set the IPL to 13, then float a ZERO thru the Interrupt Vector (bits <8:2>)',
: 0753 3 'of the UET, forcing an interrupt on level 4. After each Interrupt, check',
: 0754 3 'the VECTOR that the interrupt occurred at.',
: 0755 3 ''
: 0756 4 BEGIN
: 0757 4 LOCAL
: 0758 4 EXP_VECTOR;
: 0759 4
: 0760 4 $DS_SETIPL(LEVEL=XX'13');
: 0761 4
: 0762 4 EXP_VECTOR = XX'1F8';
: 0763 4 INCR I FROM 1 TO 7 DO
: 0764 5 BEGIN
: 0765 5 UET_DATA_REG = .EXP_VECTOR;
: 0766 5 DO ! Scope loop begins here
: 0767 6 BEGIN
: 0768 6 UET_CTRL_REG = XX'8000';
: 0769 6 $DS_WAITUS(TIME=20);
: 0770 6 INTERRUPT_EXP = 1;
: 0771 6 UET_CTRL_REG = BR4; ! Force the Interrupt
: 0772 6 $DS_WAITUS(TIME=20);
: 0773 6 IF .INT_VECTOR NEQ .EXP_VECTOR
: 0774 6 THEN
: 0775 7 BEGIN
: 0776 7 $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD,
: 0777 7 PRLINK = PRINT_GENERAL,
: 0778 7 P1 = FMT_VECTOR,P2 = 2,
: 0779 7 P3 = .EXP_VECTOR,P4 = .INT_VECTOR);
: 0780 6 $PRINT: Test 27, Subtest 5, Error 14
: 0781 6 END;
: 0782 5 WHILE $DS_INLOOP; ! Scope loop ends here
: 0783 5 EXP_VECTOR = ((.EXP_VECTOR OR 2) ^ 1) AND XX'1FC';
: 0784 4 END;
: 0785 3 END;
: 0786 3
: 0787 3
: 0788 3
: 0789 3 CODECOMMENT
: 0790 3 ''
: 0791 3 'If there are UBEs present, set their Vectors again.',
: 0792 3 ''
: 0793 4 BEGIN
: 0794 4 IF .NUM_UBE NEQ 0
: 0795 4 THEN
: 0796 4 INCR I FROM 0 TO .NUM_UBE - 1 DO
: 0797 4 $DS_SETVEC(VECTOR=.UBE_VECTOR[I],SRVADR=.UBE_SERVICE[I]);
: 0798 3 END;
: 0799 3
: 0800 2 $DS_ENDSUB;
: 0801 2
: 0802 2 $DS_CHANNEL(UNIT=.LUN,FUNC=CHC$_DSINT);
: 0803 2 ![[1.4] DS_CHANNEL moved outside DS_ENDSUB for AUTO_QA Multiple Subtest Loops.

```


ZZ-ECCBA-1.4
UBI_TESTS_27_32
01-04

TEST_027: UET CSR1 Interrupt Test
TEST_027: UET CSR1 Interrupt Test

K 3

6-Sep-1985

Fiche 3 Frame K3

Sequence 448

6-Sep-1985 17:25:18

VAX-11 Bliss-32 V4.1-746

Page 17

6-Sep-1985 17:15:07

[LEMIEUX.ECCBA.RELEASE]ECCBA8.B32;19

(3)

; 0804 2 ![[1.4] Problem was interrupts being disabled after 1st pass.
; 0805 2
; 0806 1 \$DS_ENDTEST;

.TITLE UBI_TESTS_27_32
.IDENT \01-04\

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00000 \$:
00000000 00000003 00010

.ADDRESS P.AAA, P.AAB, P.AAC, TEST_027
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

72 65 74 6E 49 20 31 52 53 43 20 54 45 55 17
74 73 65 54 20 74 70 75 72
001B 00000
00002
00011
0001A
00000000 0001C

P.AAA: .WORD 27
P.AAB: .ASCII <23>\UET CSR1 Interrupt Test\
P.AAC: .BLKB 2
.LONG 0

.PSECT \$OWN\$,NOEXE,2

00000 BYTE_PATTERN:
.BLKB 4
00004 STATUS0: .BLKB 2
00006 .BLKB 2
00008 STATUS1: .BLKB 4
0000C WORD_PATTERN:
.BLKB 10

DSA\$AL_APTMAIL= 65024
DSA\$GL_FLAGS= 65024
DSA\$GL_APTCOM= 65028
DSA\$GL_PASSES= 65032
DSA\$GL_UNITS= 65036
DSA\$GL_SECTNO= 65040
DSA\$GL_SID= 65044
DSA\$GL_MSGTYP= 65088
DSA\$GL_ERRNO= 65092
DSA\$GL_EVENT= 65096
DSA\$GL_SUBTNO= 65100
DSA\$GL_TESTNO= 65104
DSA\$GL_PASSNO= 65108
DSA\$GL_DEVLEN= 65112
DSA\$GL_DEVNAM= 65116
DSA\$GL_MSGPTR= 65128

.EXTRN BUFFER_SETUP, DECODER
.EXTRN ENCODER, MAP_BUFFERS
.EXTRN MAP_SETUP, PRINT_GENERAL
.EXTRN PROBE_ADDRESS, UNIBUS_SIZER
.EXTRN XFER_SETUP, CHAN_STAT
.EXTRN CODEWORDS, DECODED_WORDS
.EXTRN EVENT_FLAG, EXTERNAL_FLAG
.EXTRN FMT_BDP_DAT1, FMT_BDP_DAT0

ZZ-ECCBA-1.4 TEST_027: UET CSR1 Interrupt Test
UBI_TESTS_27_32
01-04 TEST_027: UET CSR1 Interrupt Test

L 3
6-Sep-1985 Fiche 3 Frame L3 Sequence 449
6-Sep-1985 17:25:18 VAX-11 Bliss-32 V4.1-746 Page 18
6-Sep-1985 17:15:07 [LEMIEUX.ECCBA.RELEASE]ECCBA8.B32;19 (3)

.EXTRN FMT_BDP_DATOB, FMT_BYTE_OFF
.EXTRN FMT_CMI_UB, FMT_CPU_RW
.EXTRN FMT_DDP_DATI, FMT_DDP_DATO
.EXTRN FMT_DDP_DATOB, FMT_DP_SEL
.EXTRN FMT_EXT_PROC, FMT_INIT_ACLO
.EXTRN FMT_MAP_ADDR, FMT_MAP_CS
.EXTRN FMT_MAP_FAIL, FMT_MAP_INV
.EXTRN FMT_MCHK, FMT_NO_ACINT
.EXTRN FMT_NO_ACINT1, FMT_NO_INT
.EXTRN FMT_NO_MCHK, FMT_NO_UBE
.EXTRN FMT_UA1, FMT_UB_CMI
.EXTRN FMT_UB_TO, FMT_UBE_DUMP
.EXTRN FMT_UBI_DUMP, FMT_UET_ACLO1
.EXTRN FMT_UET_INTD, FMT_UET_INTD2
.EXTRN FMT_UET_PB, FMT_UET_STAT
.EXTRN FMT_UNX_ACINT, FMT_UNX_ACINT1
.EXTRN FMT_UNX_ACINT2, FMT_UNX_INT1
.EXTRN FMT_VECTOR, FREE_MAP
.EXTRN HANDLER, INT_VECTOR
.EXTRN INTERRUPT_EXP, INTERRUPT_OCC
.EXTRN LUN, NOISY_WORDS
.EXTRN NUM_UBE, PRINT_CSR_ERR
.EXTRN PRINT_DCOMP_ERR, PRINT_MAP_ERR
.EXTRN UBE_BUF_ADDR, UBE_INT_OCC
.EXTRN UBE_SERVICE, UBE_VECTOR
.EXTRN UBIMOD, UBIMOD_BDP
.EXTRN UBIMOD_BYTE_OFF
.EXTRN UBIMOD_CMI, UBIMOD_CPU_RW
.EXTRN UBIMOD_CSR, UBIMOD_DDP
.EXTRN UBIMOD_DP_SEL, UBIMOD_MAP
.EXTRN UBIMOD_MAP_CS, UBIMOD_MAP_ADDR
.EXTRN UBIMOD_MAP_CTRL
.EXTRN UBIMOD_UA1, UBIMOD_UB_CMI
.EXTRN UBI_VECTOR, UBI_LINK
.EXTRN UETMOD, SCRATCH
.EXTRN UBE_BASE_ADDR, UBE_BUF_SIZE
.EXTRN UBE_DRIVE, UBE_PAGE_CNT
.EXTRN UBE_INIT_DATA, UBE_MODE
.EXTRN UBE_MAP_NO, UBE_DP_NO
.EXTRN UBE_EXP_DATA, UBE_XFER_ERR
.EXTRN UBE_STAT_ERR, UBE0_ISR
.EXTRN UBI_UNDER_TEST, UNIBUS_SPACE
.EXTRN UET_ISR, UET_ISR_2
.EXTRN VALID_MAPS, DS\$BGNSUB
.EXTRN DS\$CHANNEL, DS\$SETIPL
.EXTRN DS\$WAITUS, DS\$ERRHARD
.EXTRN DS\$INLOOP, DS\$ENDSUB
.EXTRN DS\$CLRVEC, DS\$SETVEC
.EXTRN DS\$BREAK

.PSECT TEST_027, NOWRT, 2

.ENTRY TEST_027, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 0323
R11
MOVAB PRINT_GENERAL, R11 ;

OFFC 00000

5B 0000G CF 9E 00002

```

5A      0000G  CF  9E 00007  MOVAB  INTERRUPT_OCC, R10      ;
59      0000G  CF  9E 0000C  MOVAB  LUN, R9                ;
58 00000000G  9F  9E 00011  MOVAB  a#DS$INLOOP, R8          ;
57 00000000G  9F  9E 00018  MOVAB  a#DS$ERRHARD, R7         ;
56 00000000G  9F  9E 0001F  MOVAB  a#DS$WAITUS, R6          ;
55      0000G  CF  9E 00026  MOVAB  UNIBUS_SPACE, R5         ;
54 00000000G  9F  9E 0002B  MOVAB  a#DS$SETIPL, R4           ;
                                01 DD 00032  PUSHL  #1                ; 0350
                                1B DD 00034  PUSHL  #27             ;
50 00000000G  9F          02 FB 00036  CALLS  #2, a#DS$BGNSUB      ;
65 0003F462  8F  C9 0003D  BISL3  #259170, UNIBUS_SPACE, R0    ; 0352
60      0140  8F  B0 00045  MOVW   #320, (R0)                 ; 0354
        00000000  9F  D4 0004A  CLRL   a#^X00000000           ; 0355
                                7E D4 00050  CLRL   -(SP)         ;
                                CF  9F 00052  PUSHAB CHAN_STAT      ;
                                CF  9F 00056  PUSHAB UET_ISR       ;
                                04 DD 0005A  PUSHL  #4            ;
                                69 9A 0005C  MOVZBL LUN, -(SP)     ;
00000000G  7E          05 FB 0005F  CALLS  #5, a#DS$CHANNEL      ;
                                9F          ;
;
; Set the IPL to 1F (hex), set the interrupt expected flag, clear the
; interrupt occurred flag, and initiate an interrupt at BR7. Lower the IPL
; from 1F to 17 by steps of one, checking each time that the interrupt has
; not occurred.
;
53          1F D0 00066  MOVL   #31, IPL                        ; 0366
                                1F DD 00069  1$: PUSHL  #31        ; 0368
64          01 FB 0006B  CALLS  #1, DS$SETIPL                   ;
50 65 0003F464  8F  C9 0006E  2$: BISL3  #259172, UNIBUS_SPACE, R0 ; 0370
60      8000  8F  B0 00076  MOVW   #-32768, (R0)               ; 0371
                                53 DD 0007B  PUSHL  IPL           ; 0372
64          01 FB 0007D  CALLS  #1, DS$SETIPL                   ;
0000G  CF      01 90 00080  MOVB   #1, INTERRUPT_EXP            ; 0373
65 0003F464  8F  C9 00087  CLRB   INTERRUPT_OCC                ; 0374
50 60      0800  8F  B0 0008F  BISL3  #259172, UNIBUS_SPACE, R0 ;
7E          14 7D 00094  MOVQ   #20, -(SP)                     ; 0375
66          02 FB 00097  CALLS  #2, DS$WAITUS                   ; 0377
19          6A E9 0009A  BLBC   INTERRUPT_OCC, 3$              ; 0378
;
; Interrupt occurred at IPL greater than or equal to 17
;
                                7E 7C 0009D  CLRQ   -(SP)         ; 0388
53          DD 0009F  PUSHL  IPL                                ;
07          DD 000A1  PUSHL  #7                                  ;
02          DD 000A3  PUSHL  #2                                  ;
0000G  CF  9F 000A5  PUSHAB  FMT_UNX_INT1                       ;
5B          DD 000A9  PUSHL  R11                                ;
0000G  CF  9F 000AB  PUSHAB  UBIMOD_CSR                         ;
7E          69 9A 000AF  MOVZBL LUN, -(SP)                     ;
01          DD 000B2  PUSHL  #1                                  ;
39          11 000B4  BRB     4$                                ;

```


; If this is unit 1 then check that interrupt done bit is not set

```

50      0000G  CF  D0 000B6 3$:  MOVL  UBI_LINK, R0      ; 0396
01      0B    A0  91 000BB      CMPB  11(R0), #1      ;
50      65 0003F464 8F  C9 000C1  BNEQ  5$      ;
52      60  B0 000C9      BISL3  #259172, UNIBUS_SPACE, R0 ; 0399
22      52      0E  E1 000CC      MOVW  (R0), TEMP      ;
7E      50      7E  7C 000D0      BBC   #14, TEMP, 5$      ; 0400
50      50      52  3C 000D2      CLRQ  -(SP)      ; 0405
7E      50 FFFFBFFF 8F  CB 000D5      MOVZWL TEMP, R0      ;
7E      02  7D 000DD      BICL3  #-16385, R0, -(SP)      ;
      0000G  CF  9F 000E0      MOVQ   #2, -(SP)      ;
      5B  DD 000E4      PUSHAB  FMT_UET_INTD      ;
      0000G  CF  9F 000E6      PUSHL  R11      ;
7E      69  9A 000EA      PUSHAB  UBIMOD_CSR      ;
      02  DD 000ED      MOVZBL  LUN, -(SP)      ;
67      0A  FB 000EF 4$:  CALLS  #10, DS$ERRHARD      ;
68      00  FB 000F2 5$:  CALLS  #0, DS$INLOOP      ; 0410
03      50  E9 000F5      BRW    2$      ;
      FF73 31 000F8      BLBC   R0, 6$      ;
FF67    53  FF  8F      17  9D 000FB 6$:  ACBB  #23, #-1, IPL, 1$ ; 0366

```

; Now lower the IPL to 16 and after a nominal wait check to see if the interrupt has occurred.

```

50      65 0003F464 8F  C9 00102 7$:  BISL3  #259172, UNIBUS_SPACE, R0 ; 0422
60      8000 8F  B0 0010A      MOVW  #-32768, (R0)      ; 0423
      16  DD 0010F      PUSHL  #22      ; 0424
      64      01  FB 00111      CALLS  #1, DS$SETIPL      ;
0000G  CF  01  90 00114      MOVW  #1, INTERRUPT_EXP      ; 0425
      6A  94 00119      CLRQ  INTERRUPT_OCC      ; 0426
50      65 0003F464 8F  C9 0011B      BISL3  #259172, UNIBUS_SPACE, R0 ;
60      0800 8F  B0 00123      MOVW  #2048, (R0)      ; 0427
7E      14  7D 00128      MOVQ   #20, -(SP)      ; 0428
66      02  FB 0012B      CALLS  #2, DS$WAITUS      ;
19      6A  E8 0012E      BLBS   INTERRUPT_OCC, 8$      ; 0429
      7E  7C 00131      CLRQ  -(SP)      ; 0435
      16  DD 00133      PUSHL  #22      ;
      07  DD 00135      PUSHL  #7      ;
      02  DD 00137      PUSHL  #2      ;
      00  G  CF  9F 00139      PUSHAB  FMT_NO_INT      ;
      5B  DD 0013D      PUSHL  R11      ;
      0000G  CF  9F 0013F      PUSHAB  UBIMOD_CSR      ;
7E      69  9A 00143      MOVZBL  LUN, -(SP)      ;
      03  DD 00146      PUSHL  #3      ;
      76  11 00148      BRB    11$      ;

```

; If this is unibus number 1 check that the interrupt done bit in CSR1 is set. Then check that it does not clear when it is written to a zero and that it does clear when written to a one.

```

50      0000G  CF  D0 0014A 8$:  MOVL  UBI_LINK, R0      ; 0446
01      0B    A0  91 0014F      CMPB  11(R0), #1      ;
                                03  13 00153      BEQL  9$      ;
                                00A0 31 00155      BRW   14$      ;
50      65 0003F464 8F  C9 00158 9$:  BISL3  #259172, UNIBUS_SPACE, R0 ; 0449
52      60      60  B0 00160      MOVW   (R0), TEMP      ;
25      52      0E  E0 00163      BBS    #14, TEMP, 10$    ; 0450
                                ; Interrupt done bit in CSR1 did not set.
                                ;
                                7E  7C 00167      CLRQ   -(SP)      ; 0460
50      52      52  3C 00169      MOVZWL  TEMP, R0      ;
7E      50 FFFFBFFF 8F  CB 0016C      BICL3   #-16385, R0, -(SP) ;
7E      4000      8F  3C 00174      MOVZWL  #16384, -(SP) ;
                                02  DD 00179      PUSHL   #2      ;
                                0000G CF  9F 0017B      PUSHAB  FMT_UET_INTD ;
                                5B  DD 0017F      PUSHL   R11      ;
                                0000G CF  9F 00181      PUSHAB  UBIMOD_CSR ;
7E      69  9A 00185      MOVZBL  LUN, -(SP) ;
                                04  DD 00188      PUSHL   #4      ;
                                69  11 0018A      BRB    13$      ;
                                ; Check that Interrupt Done bit clears properly. First
                                ; try and clear it by writting a zero and check that
                                ; that it does not clear.
                                ;
50      65 0003F464 8F  C9 0018C 10$:  BISL3  #259172, UNIBUS_SPACE, R0 ; 0469
52      60      60  B4 00194      CLRW   (R0)      ; 0470
25      52      60  B0 00196      MOVW   (R0), TEMP      ; 0471
52      0E  E0 00199      BBS    #14, TEMP, 12$    ; 0472
                                ; Interrupt done bit cleared when writting a zero.
                                ;
                                7E  7C 0019D      CLRQ   -(SP)      ; 0482
50      52      52  3C 0019F      MOVZWL  TEMP, R0      ;
7E      50 FFFFBFFF 8F  CB 001A2      BICL3   #-16385, R0, -(SP) ;
7E      4000      8F  3C 001AA      MOVZWL  #16384, -(SP) ;
                                02  DD 001AF      PUSHL   #2      ;
                                0000G CF  9F 001B1      PUSHAB  FMT_UET_INTD ;
                                5B  DD 001B5      PUSHL   R11      ;
                                0000G CF  9F 001B7      PUSHAB  UBIMOD_CSR ;
7E      69  9A 001BB      MOVZBL  LUN, -(SP) ;
                                05  DD 001BE      PUSHL   #5      ;
                                33  11 001C0 11$:  BRB    13$      ;
                                ; Check that it clears when a 1 is written to it.
                                ;
50      65 0003F464 8F  C9 001C2 12$:  BISL3  #259172, UNIBUS_SPACE, R0 ; 0489
52      60      60  B0 001CA      MOVW   #16384, (R0) ; 0490
52      60  B0 001CF      MOVW   (R0), TEMP ; 0491
22      52      0E  E1 001D2      BBC     #14, TEMP, 14$ ; 0492
                                7E  7C 001D6      CLRQ   -(SP) ; 0497
50      52      52  3C 001D8      MOVZWL  TEMP, R0      ;
7E      50 FFFFBFFF 8F  CB 001DB      BICL3   #-16385, R0, -(SP) ;

```

```

7E      0000G 02 7D 001E3      MOVQ    #2, -(SP)
          CF 9F 001E6      PUSHAB   FMT_UET_INTD
          5B DD 001EA      PUSHL    R11
          0000G CF 9F 001EC      PUSHAB   UBIMOD_CSR
7E      69 9A 001F0      MOVZBL   LUN, -(SP)
          06 DD 001F3      PUSHL    #6
67      0A FB 001F5 13$:      CALLS    #10, DS$ERRHARD
68      00 FB 001F8 14$:      CALLS    #0, DS$INLOOP
03      50 E9 001FB      BLBC     R0, 15$
          FF01 31 001FE      BRW      7$
          01 DD 00201 15$:      PUSHL    #1
          1B DD 00203      PUSHL    #27
00000000G 9F      02 FB 00205      CALLS    #2, a#DS$ENDSUB
          02 DD 0020C      PUSHL    #2
          1B DD 0020E      PUSHL    #27
00000000G 9F      02 FB 00210      CALLS    #2, a#DS$BGNSUB
;
; Set the IPL to 1F (hex), set the interrupt expected flag, clear the
; interrupt occurred flag, and initiate an interrupt at BR6. Lower the IPL
; from 1F to 16 by steps of one, checking each time that the interrupt has
; not occurred.
;
52      1F D0 00217      MOVL     #31, IPL
          1F DD 0021A 16$:      PUSHL    #31
64      01 FB 0021C      CALLS    #1, DS$SETIPL
50      65 0003F464 17$:      BISL3    #259172, UNIBUS_SPACE, R0
60      8000      8F B0 00227      MOVW     #-32768, (R0)
          52 DD 0022C      PUSHL    IPL
          64 01 FB 0022E      CALLS    #1, DS$SETIPL
          0000G CF 01 90 00231      MOVB     #1, INTERRUPT_EXP
          6A 94 00236      CLRB     INTERRUPT_OCC
50      65 0003F464 8F C9 00238      BISL3    #259172, UNIBUS_SPACE, R0
60      0400      8F B0 00240      MOVW     #1024, (R0)
7E      14 7D 00245      MOVQ     #20, -(SP)
66      02 FB 00248      CALLS    #2, DS$WAITUS
1A      6A E9 0024B      BLBC     INTERRUPT_OCC, 18$
          7E 7C 0024E      CLRQ     -(SP)
          52 DD 00250      PUSHL    IPL
          06 DD 00252      PUSHL    #6
          02 DD 00254      PUSHL    #2
          0000G CF 9F 00256      PUSHAB   FMT_UNX_INT1
          5B DD 0025A      PUSHL    R11
          0000G CF 9F 0025C      PUSHAB   UBIMOD_CSR
7E      69 9A 00260      MOVZBL   LUN, -(SP)
          07 DD 00263      PUSHL    #7
67      0A FB 00265      CALLS    #10, DS$ERRHARD
68      00 FB 00268 18$:      CALLS    #0, DS$INLOOP
B1      50 E8 0026B      BLBS     R0, 17$
FFA5      52 FF 8F      16 9D 0026E      ACBB     #22, #-1, IPL, 16$

```

; Now lower the IPL to 15 and after a nominal wait check to see if the

; interrupt has occurred.

50	65	0003F464	8F	C9	00275	19\$:	BISL3	#259172, UNIBUS_SPACE, R0	:	0552
	60	8000	8F	B0	0027D		MOVW	#-32768, (R0)	:	0553
	0000G	CF	01	90	00282		MOVB	#1, INTERRUPT_EXP	:	0554
			6A	94	00287		CLRB	INTERRUPT_OCC	:	0555
			15	DD	00289		PUSHL	#21	:	0556
	64		01	FB	0028B		CALLS	#1, DS\$SETIPL	:	
50	65	0003F464	8F	C9	0028E		BISL3	#259172, UNIBUS_SPACE, R0	:	
	60	0400	8F	B0	00296		MOVW	#1024, (R0)	:	0557
	7E		14	7D	0029B		MOVQ	#20, -(SP)	:	0558
	66		02	FB	0029E		CALLS	#2, DS\$WAITUS	:	
	1A		6A	E8	002A1		BLBS	INTERRUPT_OCC, 20\$	:	0559
			7E	7C	002A4		CLRQ	-(SP)	:	0564
			0F	DD	002A6		PUSHL	#15	:	
			06	DD	002A8		PUSHL	#6	:	
			02	DD	002AA		PUSHL	#2	:	
		0000G	CF	9F	002AC		PUSHAB	FMT_NO_INT	:	
			5B	DD	002B0		PUSHL	R11	:	
		0000G	CF	9F	002B2		PUSHAB	UBIMOD_CSR	:	
	7E		69	9A	002B6		MOVZBL	LUN, -(SP)	:	
			08	DD	002B9		PUSHL	#8	:	
	67		0A	FB	002BB		CALLS	#10, DS\$ERRHARD	:	
	68		00	FB	002BE	20\$:	CALLS	#0, DS\$INLOOP	:	0566
	B1		50	E8	002C1		BLBS	R0, 19\$	:	
			02	DD	002C4		PUSHL	#2	:	0568
			1B	DD	002C6		PUSHL	#27	:	
	00000000G	9F	02	FB	002C8		CALLS	#2, a#DS\$ENDSUB	:	
			03	DD	002CF		PUSHL	#3	:	0570
			1B	DD	002D1		PUSHL	#27	:	
	00000000G	9F	02	FB	002D3		CALLS	#2, a#DS\$BGNSUB	:	

; Set the IPL to 1F (hex), set the interrupt expected flag, clear the
; interrupt occurred flag, and initiate an interrupt at BR5. Lower the IPL
; from 1F to 15 by steps of one, checking each time that the interrupt has
; not occurred.

	52		1F	D0	002DA		MOVL	#31, IPL	:	0583
			1F	DD	002DD	21\$:	PUSHL	#31	:	0585
	64		01	FB	002DF		CALLS	#1, DS\$SETIPL	:	
50	65	0003F464	8F	C9	002E2	22\$:	BISL3	#259172, UNIBUS_SPACE, R0	:	0587
	60	8000	8F	B0	002EA		MOVW	#-32768, (R0)	:	0588
			52	DD	002EF		PUSHL	IPL	:	0589
	64		01	FB	002F1		CALLS	#1, DS\$SETIPL	:	
	0000G	CF	01	90	002F4		MOVB	#1, INTERRUPT_EXP	:	0590
			6A	94	002F9		CLRB	INTERRUPT_OCC	:	0591
50	65	0003F464	8F	C9	002FB		BISL3	#259172, UNIBUS_SPACE, R0	:	
	60	0200	8F	B0	00303		MOVW	#512, (R0)	:	0592
	7E		14	7D	00308		MOVQ	#20, -(SP)	:	0594
	66		02	FB	0030B		CALLS	#2, DS\$WAITUS	:	
	1A		6A	E9	0030E		BLBC	INTERRUPT_OCC, 23\$	:	0595
			7E	7C	00311		CLRQ	-(SP)	:	0600

Address	Hex	Label	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	
---------	-----	-------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--

```

      52      1F D0 0039D      MOVL      #31, IPL                      ; 0646
      1F DD 003A0 26$:      PUSHL     #31                          ; 0648
      64      01 FB 003A2      CALLS    #1, DS$SETIPL                ;
      50      65 0003F464 8F C9 003A5 27$:      BISL3    #259172, UNIBUS_SPACE, R0 ; 0650
      60      8000      8F B0 003AD      MOVW      #-32768, (R0)      ; 0651
      52      DD 003B2      PUSHL     IPL                          ; 0652
      64      01 FB 003B4      CALLS    #1, DS$SETIPL                ;
      0000G   CF      01 90 003B7      MOVW      #1, INTERRUPT_EXP      ; 0653
      6A      94 003BC      CLRB      INTERRUPT_OCC                ; 0654
      50      65 0003F464 8F C9 003BE      BISL3    #259172, UNIBUS_SPACE, R0 ;
      60      0100      8F B0 003C6      MOVW      #256, (R0)          ; 0655
      7E      14 7D 003CB      MOVQ      #20, -(SP)                  ; 0657
      66      02 FB 003CE      CALLS    #2, DS$WAITUS                ;
      1A      6A E9 003D1      BLBC      INTERRUPT_OCC, 28$          ; 0658
      7E      7C 003D4      CLRQ      -(SP)                          ; 0663
      52      DD 003D6      PUSHL     IPL                          ;
      04      DD 003D8      PUSHL     #4                            ;
      02      DD 003DA      PUSHL     #2                            ;
      0000G   CF      9F 003DC      PUSHAB   FMT_UNX_INT1            ;
      5B      DD 003E0      PUSHL     R11                          ;
      0000G   CF      9F 003E2      PUSHAB   UBIMOD_CSR              ;
      7E      69 9A 003E6      MOVZBL   LUN, -(SP)                  ;
      0B      DD 003E9      PUSHL     #11                          ;
      67      0A FB 003EB      CALLS    #10, DS$ERRHARD                ;
      68      00 FB 003EE 28$:      CALLS    #0, DS$INLOOP            ; 0666
      B1      50 E8 003F1      BLBS      R0, 27$                    ;
      FFA5    52      FF 8F      14 9D 003F4      ACBB      #20, #-1, IPL, 26$ ; 0646
;
; Now lower the IPL to 13 and after a nominal wait check to see
; if the
; interrupt has occurred.
;
      50      65 0003F464 8F C9 003FB 29$:      BISL3    #259172, UNIBUS_SPACE, R0 ; 0678
      60      8000      8F B0 00403      MOVW      #-32768, (R0)      ; 0679
      0000G   CF      01 90 00408      MOVW      #1, INTERRUPT_EXP      ; 0680
      6A      94 0040D      CLRB      INTERRUPT_OCC                ; 0681
      13      DD 0040F      PUSHL     #19                          ; 0682
      64      01 FB 00411      CALLS    #1, DS$SETIPL                ;
      50      65 0003F464 8F C9 00414      BISL3    #259172, UNIBUS_SPACE, R0 ;
      60      0100      8F B0 0041C      MOVW      #256, (R0)          ; 0683
      7E      14 7D 00421      MOVQ      #20, -(SP)                  ; 0684
      66      02 FB 00424      CALLS    #2, DS$WAITUS                ;
      1A      6A E8 00427      BLBS      INTERRUPT_OCC, 30$          ; 0685
      7E      7C 0042A      CLRQ      -(SP)                          ; 0690
      0D      DD 0042C      PUSHL     #13                          ;
      04      DD 0042E      PUSHL     #4                            ;
      02      DD 00430      PUSHL     #2                            ;
      0000G   CF      9F 00432      PUSHAB   FMT_NO_INT              ;
      5B      DD 00436      PUSHL     R11                          ;
      0000G   CF      9F 00438      PUSHAB   UBIMOD_CSR              ;
      7E      69 9A 0043C      MOVZBL   LUN, -(SP)                  ;
      0C      DD 0043F      PUSHL     #12                          ;
      67      0A FB 00441      CALLS    #10, DS$ERRHARD                ;
      68      00 FB 00444 30$:      CALLS    #0, DS$INLOOP            ; 0692
      B1      50 E8 00447      BLBS      R0, 29$                    ;

```



```

                                04 DD 0044A    PUSHL    #4                ; 0694
                                1B DD 0044C    PUSHL    #27             ;
00000000G 9F                   02 FB 0044E    CALLS    #2, a#DS$ENDSUB    ;
                                05 DD 00455    PUSHL    #5                ; 0696
                                1B DD 00457    PUSHL    #27             ;
00000000G 9F                   02 FB 00459    CALLS    #2, a#DS$BGNSUB    ;
                                ;
                                ; If there are UBEs selected, release their vectors.
                                ;
                                53 0000G CF D0 00460    MOVL     NUM_UBE, R3        ; 0708
                                16 13 00465    BEQL     33$                  ;
                                52 01 CE 00467    MNEGL   #1, I                ; 0710
                                0D 11 0046A    BRB      32$                  ;
                                7E 0000GCF42 3C 0046C 31$: MOVZWL  UBE_VECTOR[I], -(SP)    ; 0711
EF 00000000G 9F 01 FB 00472    CALLS    #1, a#DS$CLRVEC    ;
                                52 53 F2 00479 32$:  AOBLS   R3, I, 31$        ;
                                ;
                                ; Set the IPL to 13, then float a ONE thru the Interrupt Vector
                                ; (bits <8:2>)
                                ; of the UET, forcing an interrupt on level 4. After each Inter
                                ; rupt, check
                                ; the VECTOR that the interrupt occurred at.
                                ;
                                13 DD 0047D 33$:  PUSHL    #19                ; 0724
                                01 FB 0047F    CALLS    #1, DS$SETIPL        ;
                                52 02 D0 00482    MOVL     #2, I                ; 0726
                                53 01 78 00485 34$:  ASHL     I, #1, EXP_VECTOR    ; 0728
                                50 65 0003F462 8F C9 00489    BISL3   #259170, UNIBUS_SPACE, R0    ;
                                60 53 B0 00491    MOVW     EXP_VECTOR, (R0)        ; 0729
                                50 65 0003F464 8F C9 00494 35$:  BISL3   #259172, UNIBUS_SPACE, R0    ; 0731
                                60 8000 8F B0 0049C    MOVW     #-32768, (R0)        ; 0732
                                7E 14 7D 004A1    MOVQ     #20, -(SP)            ; 0733
                                66 02 FB 004A4    CALLS    #2, DS$WAITUS        ;
                                50 0000G CF 01 90 004A7    MOVB     #1, INTERRUPT_EXP    ; 0734
                                65 0003F464 8F C9 004AC    BISL3   #259172, UNIBUS_SPACE, R0    ;
                                60 0100 8F B0 004B4    MOVW     #256, (R0)        ; 0735
                                7E 14 7D 004B9    MOVQ     #20, -(SP)            ; 0736
                                66 02 FB 004BC    CALLS    #2, DS$WAITUS        ;
                                53 0000G CF D1 004BF    CMPL     INT_VECTOR, EXP_VECTOR    ; 0737
                                1C 13 004C4    BEQL     36$                  ;
                                7E 7C 004C6    CLRQ     -(SP)                ; 0743
                                0000G CF DD 004C8    PUSHL    INT_VECTOR        ;
                                53 DD 004CC    PUSHL    EXP_VECTOR        ;
                                02 DD 004CE    PUSHL    #2                  ;
                                0000G CF 9F 004D0    PUSHAB   FMT_VECTOR        ;
                                5B DD 004D4    PUSHL    R11                  ;
                                0000G CF 9F 004D6    PUSHAB   UBIMOD            ;
                                7E 69 9A 004DA    MOVZBL   LUN, -(SP)            ;
                                0D DD 004DD    PUSHL    #13                  ;
                                67 0A FB 004DF    CALLS    #10, DS$ERRHARD        ;
                                68 00 FB 004E2 36$:  CALLS    #0, DS$INLOOP        ; 0746
                                AC 50 E8 004E5    BLBS     R0, 35$            ;
                                99 52 08 F3 004E8    AOBLEQ   #8, I, 34$        ; 0726
                                ;
                                ; Set the IPL to 13, then float a ZERO thru the Interrupt Vecto

```

;
;

r (bits <8:2>)
; of the UET, forcing an interrupt on level 4. After each Inter
rupt, check
; the VECTOR that the interrupt occurred at.

		13	DD	004EC	PUSHL	#19		; 0760
	64	01	FB	004EE	CALLS	#1, DS\$SETIPL		
	52	8F	3C	004F1	MOVZWL	#504, EXP_VECTOR		; 0762
	53	01	D0	004F6	MOVL	#1, I		; 0763
50	65	0003F462	8F	C9 004F9	37\$: BISL3	#259170, UNIBUS_SPACE, R0		; 0764
	60		52	B0 00501	MOVW	EXP_VECTOR, (R0)		; 0765
50	65	0003F464	8F	C9 00504	38\$: BISL3	#259172, UNIBUS_SPACE, R0		; 0767
	60	8000	8F	B0 0050C	MOVW	#-32768, (R0)		; 0768
	7E		14	7D 00511	MOVQ	#20, -(SP)		; 0769
	66		02	FB 00514	CALLS	#2, DS\$WAITUS		
	CF	0000G	01	90 00517	MOVB	#1, INTERRUPT_EXP		; 0770
50	65	0003F464	8F	C9 0051C	BISL3	#259172, UNIBUS_SPACE, R0		
	60	0100	8F	B0 00524	MOVW	#256, (R0)		; 0771
	7E		14	7D 00529	MOVQ	#20, -(SP)		; 0772
	66		02	FB 0052C	CALLS	#2, DS\$WAITUS		
	52	0000G	CF	D1 0052F	CMPL	INT_VECTOR, EXP_VECTOR		; 0773
			1C	13 00534	BEQL	39\$		
			7E	7C 00536	CLRQ	-(SP)		; 0779
		0000G	CF	DD 00538	PUSHL	INT_VECTOR		
			52	DD 0053C	PUSHL	EXP_VECTOR		
			02	DD 0053E	PUSHL	#2		
		0000G	CF	9F 00540	PUSHAB	FMT_VECTOR		
			5B	DD 00544	PUSHL	R11		
		0000G	CF	9F 00546	PUSHAB	UBIMOD		
	7E		69	9A 0054A	MOVZBL	LUN, -(SP)		
			0E	DD 0054D	PUSHL	#14		
	67		0A	FB 0054F	CALLS	#10, DS\$ERRHARD		
	68		00	FB 00552	39\$: CALLS	#0, DS\$INLOOP		; 0782
	AC		50	E8 00555	BLBS	R0, 38\$		
50	52		01	78 00558	ASHL	#1, EXP_VECTOR, R0		; 0783
	50	FFFFFFE03	8F	CA 0055C	BICL2	#-509, R0		
52	50		04	C9 00563	BISL3	#4, R0, EXP_VECTOR		
8E	53		07	F3 00567	AOBLEQ	#7, I, 37\$		; 0763

; If there are UBEs present, set their Vectors again.

	53	0000G	CF	D0 0056B	MOVL	NUM_UBE, R3		; 0794
			1D	13 00570	BEQL	42\$		
	52		01	CE 00572	MNEGL	#1, I		; 0796
			14	11 00575	BRB	41\$		
			7E	D4 00577	40\$: CLRL	-(SP)		; 0797
		0000GCF	42	DD 00579	PUSHL	UBE_SERVICE[I]		
	7E	0000GCF	42	3C 0057E	MOVZWL	UBE_VECTOR[I], -(SP)		
	9F		03	FB 00584	CALLS	#3, @DS\$SETVEC		
E8	52		53	F2 0058B	41\$: AOBLSS	R3, I, 40\$		
			05	DD 0058F	42\$: PUSHL	#5		; 0798
			1B	DD 00591	PUSHL	#27		
		00000000G	9F	02 FB 00593	CALLS	#2, @DS\$ENDSUB		
			9F	D4 0059A	CLRL	@#^X00000000		; 0802
			7E	7C 005A0	CLRQ	-(SP)		

ZZ-ECCBA-1.4
UBI_TESTS_27_32
01-04

TEST_027: UET CSR1 Interrupt Test
TEST_027: UET CSR1 Interrupt Test

I 4
6-Sep-1985
6-Sep-1985 17:25:18
6-Sep-1985 17:15:07

Fiche 3
Frame 14
VAX-11 Bliss-32 V4.1-746
[LEMIEUX.ECCBA.RELEASE]ECCBA8.B32;19

Sequence 459

Page 28
(3)

	7E	05	7D	005A2	MOVQ	#5, -(SP)	;
	7E	69	9A	005A5	MOVZBL	LUN, -(SP)	;
00000000G	9F	05	FB	005A8	CALLS	#5, a#DS\$CHANNEL	;
00000000G	9F	00	FB	005AF	CALLS	#0, a#DS\$BREAK	;
	50	01	D0	005B6	MOVL	#1, R0	; 0806
			04	005B9	RET		;

; Routine Size: 1466 bytes, Routine Base: TEST_027 + 0000


```

: 0807 2 $DS_BGNTST (SECTION = (DEFAULT,ALL), TEXT = 'UET CSR2 Interrupt Test');
: 0808 2
: 0809 2 !++
: 0810 2 ! TEST DESCRIPTION
: 0811 2 !
: 0812 2 ! This test is executed only for the second unibus and only if
: 0813 2 ! it is not in external processor mode. This test will verify the
: 0814 2 ! ability of the interrupt logic in CSR2 to interrupt the
: 0815 2 ! 11/750 CPU. If external processor mode is enabled, these interrupts
: 0816 2 ! are directed to the external processor rather than the 11/750.
: 0817 2 !
: 0818 2 ! SUBTEST 1
: 0819 2 !
: 0820 2 ! First, this subtest verifys that that interrupt does not occur
: 0821 2 ! if the IPL level is 17(X) or higher. Then, the IPL level is
: 0822 2 ! lowered to 16(X). The interrupt should occur. Then, CR2 bit 14
: 0823 2 ! is verified to be ascerted. Then, CR2 bit 14 is verified not to
: 0824 2 ! clear when writting a zero to it and that it does clear when
: 0825 2 ! writting a one to it.
: 0826 2 !
: 0827 2 ! SUBTEST 2, 3, and 4
: 0828 2 !
: 0829 2 ! These subtests ensure that interrupts at BR6, BR5, and BR4
: 0830 2 ! function correctly.
: 0831 2 !
: 0832 2 ! SUBTEST 5
: 0833 2 !
: 0834 2 ! This subtest ensures that the interrupt vector is correct by
: 0835 2 ! floating a one and a zero thru the vector.
: 0836 2 !
: 0837 2 ! --
: 0838 2
: 0839 2 REGISTER
: 0840 2 IPL,
: 0841 2 TEMP: WORD,
: 0842 2 I;
: 0843 2
: 0844 2 IF .UBI_LINK[HP$B_DRIVE] NEQ 1
: 0845 2 THEN $DS_ABORT(ARG = TEST)
: 0846 2 ELSE
: 0847 3 BEGIN
: 0848 3 TEMP = .UET_CTRL_REG2;
: 0849 3 IF (.TEMP AND %X'8000') EQL %X'8000'
: 0850 3 THEN
: 0851 4 BEGIN
: 0852 4
: 0853 4 !
: 0854 4 ! Only print this message if pass 1.
: 0855 4 !
: 0856 5 IF ((.DSA$GL_PASSNO LSS 2) AND (.EXTERNAL_FLAG EQL 0))
: 0857 4 THEN
: 0858 5 BEGIN
: 0859 5 $DS_PRINTF(FMT_EXT_PROC);
: 0860 5 EXTERNAL_FLAG = 1;
: 0861 4 END;

```

```

: 0862 4      $DS_ABORT(ARG = TEST);
: 0863 3      END;
: 0864 2      END;
: 0865 2
: 0866 3      $DS_BGNSUB;                ! Subtest 1: BR7
: 0867 3
: 0868 3      CODECOMMENT
: 0869 3      ``
: 0870 3      'If there are UBEs selected, release their vectors.',
: 0871 3      ``
: 0872 4      BEGIN
: 0873 4      IF .NUM_UBE NEQ 0
: 0874 4      THEN
: 0875 4          INCR I FROM 0 TO .NUM_UBE - 1 DO
: 0876 4              $DS_CLRVEC(VECTOR=.UBE_VECTOR[I]);
: 0877 3      END;
: 0878 3      $DS_CHANNEL(UNIT=.LUN,FUNC=CHC$_ENINT,VECADR=UET_ISR_2,STSADR=CHAN_STAT);
: 0879 3      UET_CTRL_REG2 = %X'140';
: 0880 3
: 0881 3      CODECOMMENT
: 0882 3      ``
: 0883 3      'Set the IPL to 1F (hex), set the interrupt expected flag, clear the',
: 0884 3      'interrupt occurred flag, and initiate an interrupt at BR7. Lower the IPL',
: 0885 3      'from 1F to 17 by steps of one, checking each time that the interrupt has',
: 0886 3      'not occurred.',
: 0887 3      ``
: 0888 4      BEGIN
: 0889 4
: 0890 4      DECR IPL FROM %X'1F' TO %X'17' DO
: 0891 5      BEGIN
: 0892 5      $DS_SETIPL(LEVEL=%X'1F');
: 0893 5      DO                                ! Scope loop starts here
: 0894 6      BEGIN
: 0895 6      UET_CTRL_REG = %X'8000';
: 0896 6      $DS_SETIPL(LEVEL=.IPL);
: 0897 6      INTERRUPT_EXP = 1;
: 0898 6      INTERRUPT_OCC = 0;
: 0899 6      TEMP = .UET_CTRL_REG2;
: 0900 6      UET_CTRL_REG2 = (.TEMP AND %X'1FF') OR BR7 * 2;
: 0901 6
: 0902 6      $DS_WAITUS(TIME=20);                ! A8
: 0903 6      IF .INTERRUPT_OCC
: 0904 6      THEN
: 0905 6      CODECOMMENT
: 0906 6      ``
: 0907 6      'Interrupt occurred at IPL greater than or equal to 17',
: 0908 6      ``
: 0909 7      BEGIN
: 0910 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: 0911 7      PRLINK = PRINT_GENERAL,
: 0912 7      P1 = FMT_UNX_INT1,P2=2,
: 0913 7      P3 = 7,P4 = .IPL);
: 0914 7      Test 28, Subtest 1, Error 1
: 0915 6      ELSE

```

```

: 0916 6          CODECOMMENT
: 0917 6          ''
: 0918 6          'Check that the interrupt done bit in CSR2 is not set',
: 0919 6          ''
: 0920 7          BEGIN
: 0921 7          TEMP = .UET_CTRL_REG2;
: 0922 7          IF (.TEMP AND %X'4000') NEQ 0
: 0923 7          THEN
: P 0924 7          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0925 7          PRLINK = PRINT_GENERAL,
: P 0926 7          P1 = FMT_UET_INTD2,P2 = 2,
: L 0927 7          P3 = 0,P4 = .TEMP AND %X'4000');
: %PRINT:          Test 28, Subtest 1, Error 2
: 0928 6          END;
: 0929 6
: 0930 6          END
: 0931 5          WHILE $DS_INLOOP;          ! Scope loop ends here
: 0932 4          END;
: 0933 3          END;
: 0934 3
: 0935 3          CODECOMMENT
: 0936 3          ''
: 0937 3          'Now lower the IPL to 16 and after a nominal wait check to see if the',
: 0938 3          'interrupt has occurred.',
: 0939 3          ''
: 0940 4          BEGIN
: 0941 4
: 0942 4          DO          ! Scope loop begins here
: 0943 5          BEGIN
: 0944 5          UET_CTRL_REG = %X'8000';
: 0945 5          INTERRUPT_EXP = 1;
: 0946 5          INTERRUPT_OCC = 0;
: 0947 5          $DS_SETIPL(LEVEL=%X'16');
: 0948 5          TEMP = .UET_CTRL_REG2;
: 0949 5          UET_CTRL_REG2 = (.TEMP AND %X'1FF') OR BR7 * 2;
: 0950 5          $DS_WAITUS(TIME=20);
: 0951 5          IF NOT .INTERRUPT_OCC
: 0952 5          THEN
: 0953 5          CODECOMMENT
: 0954 5          ''
: 0955 5          'Interrupt did not occur.',
: 0956 5          ''
: 0957 6          BEGIN
: P 0958 6          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0959 6          PRLINK = PRINT_GENERAL,
: P 0960 6          P1 = FMT_NO_INT,P2 = 2,
: L 0961 6          P3 = 7,P4 = %X'16');
: %PRINT:          Test 28, Subtest 1, Error 3
: 0962 6          END
: 0963 5          ELSE
: 0964 5          CODECOMMENT
: 0965 5          ''
: 0966 5          'Check that the interrupt done bit in CSR2 is set.',
: 0967 5          ''
: 0968 6          BEGIN

```



```

: 0969 6      TEMP = .UET_CTRL_REG2;
: 0970 6      IF (.TEMP AND %X'4000') NEQ %X'4000'
: 0971 6      THEN
: P 0972 6          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0973 6              PRLINK = PRINT_GENERAL,
: P 0974 6              P1 = FMT_UET_INTD2,P2 = 2,
: L 0975 6              P3 = %X'4000',P4 = .TEMP AND %X'4000')
: %PRINT:      Test 28, Subtest 1, Error 4
: 0976 6      ELSE
: 0977 6          CODECOMMENT
: 0978 6          ..
: 0979 6          'Check that the interrupt done bit clears properly.',
: 0980 6          'First try and clear it by writting a zero and check',
: 0981 6          'that it does not clear.',
: 0982 6          ..
: 0983 7          BEGIN
: 0984 7          UET_CTRL_REG2 = 0; ! Try and clear by writting 0
: 0985 7          TEMP = .UET_CTRL_REG2;
: 0986 7          IF (.TEMP AND %X'4000') NEQ %X'4000'
: 0987 7          THEN
: P 0988 7              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 0989 7                  PRLINK=PRINT_GENERAL,
: P 0990 7                  P1=FMT_UET_INTD2,P2=2,
: L 0991 7                  P3=%X'4000',P4=.TEMP AND %X'4000')
: %PRINT:      Test 28, Subtest 1, Error 5
: 0992 7      ELSE
: 0993 7          CODECOMMENT
: 0994 7          ..
: 0995 7          'Now write a one to it and check that it clears',
: 0996 7          ..
: 0997 8          BEGIN
: 0998 8          UET_CTRL_REG2 = %X'4000';
: 0999 8          TEMP = .UET_CTRL_REG2;
: 1000 8          IF (.TEMP AND %X'4000') NEQ 0
: 1001 8          THEN
: P 1002 8              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: P 1003 8                  PRLINK=PRINT_GENERAL,
: P 1004 8                  P1=FMT_UET_INTD2,P2=2,
: L 1005 8                  P3=0,P4=.TEMP AND %X'4000');
: %PRINT:      Test 28, Subtest 1, Error 6
: 1006 7      END;
: 1007 6      END;
: 1008 5      END;
: 1009 5      END
: 1010 4      WHILE $DS_INLOOP;          ! Scope loop ends here
: 1011 4      END;
: 1012 3
: 1013 3
: 1014 2      $DS_ENDSUB;                ! End of Subtest 1
: 1015 2
: 1016 3      $DS_BGNSUB;                ! Subtest 2: BR6
: 1017 3
: 1018 3      CODECOMMENT
: 1019 3      ..
: 1020 3      'Set the IPL to 1F (hex), set the interrupt expected flag, clear the',

```

```

; 1021 3 'interrupt occurred flag, and initiate an interrupt at BR6. Lower the IPL',
; 1022 3 'from 1F to 16 by steps of one, checking each time that the interrupt has',
; 1023 3 'not occurred.',
; 1024 3 '':
; 1025 4 BEGIN
; 1026 4
; 1027 4 DECR IPL FROM %X'1F' TO %X'16' DO
; 1028 5 BEGIN
; 1029 5 $DS_SETIPL(LEVEL=%X'1F');
; 1030 5 DO ! Scope loop starts here
; 1031 6 BEGIN
; 1032 6 UET_CTRL_REG = %X'8000';
; 1033 6 $DS_SETIPL(LEVEL=.IPL);
; 1034 6 INTERRUPT_EXP = 1;
; 1035 6 INTERRUPT_OCC = 0;
; 1036 6 TEMP = .UET_CTRL_REG2;
; 1037 6 UET_CTRL_REG2 = (.TEMP AND %X'1FF') OR BR6 * 2;
; 1038 6
; 1039 6 $DS_WAITUS(TIME=20); ! A8
; 1040 6 IF .INTERRUPT_OCC
; 1041 6 THEN
; P 1042 6 $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; P 1043 6 PRLINK = PRINT_GENERAL,
; P 1044 6 P1 = FMT_UNX_INT1,P2 = 2,
; L 1045 6 P3 = 6,P4 = .IPL);
; xPRINT: Test 28, Subtest 2, Error 7
; 1046 6
; 1047 6 END
; 1048 5 WHILE $DS_INLOOP; ! Scope loop ends here
; 1049 4 END;
; 1050 3 END;
; 1051 3
; 1052 3 CODECOMMENT
; 1053 3 '':
; 1054 3 'Now lower the IPL to 15 and after a nominal wait check to see if the',
; 1055 3 'interrupt has occurred.',
; 1056 3 '':
; 1057 4 BEGIN
; 1058 4
; 1059 4 DO ! Scope loop begins here
; 1060 5 BEGIN
; 1061 5 UET_CTRL_REG = %X'8000';
; 1062 5 INTERRUPT_EXP = 1;
; 1063 5 INTERRUPT_OCC = 0;
; 1064 5 $DS_SETIPL(LEVEL=%X'15');
; 1065 5 TEMP = .UET_CTRL_REG2;
; 1066 5 UET_CTRL_REG2 = (.TEMP AND %X'1FF') OR BR6 * 2;
; 1067 5 $DS_WAITUS(TIME=20);
; 1068 5 IF NOT .INTERRUPT_OCC
; 1069 5 THEN
; P 1070 5 $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; P 1071 5 PRLINK = PRINT_GENERAL,
; P 1072 5 P1 = FMT_NO_INT,P2 = 2,
; L 1073 5 P3 = 6,P4 = 15);
; xPRINT: Test 28, Subtest 2, Error 8

```

```

: 1074 5      END
: 1075 4      WHILE $DS_INLOOP;          ! Scope loop ends here
: 1076 4
: 1077 3      END;
: 1078 3
: 1079 2      $DS_ENDSUB;                ! End of Subtest 2
: 1080 2
: 1081 3      $DS_BGNSUB;                ! Subtest 3: BR5
: 1082 3
: 1083 3      CODECOMMENT
: 1084 3      'Set the IPL to 1F (hex), set the interrupt expected flag, clear the',
: 1085 3      'interrupt occurred flag, and initiate an interrupt at BR5. Lower the IPL',
: 1086 3      'from 1F to 15 by steps of one, checking each time that the interrupt has',
: 1087 3      'not occurred.',
: 1088 3      '':
: 1089 3      BEGIN
: 1090 4          DECR IPL FROM $X'1F' TO $X'15' DO
: 1091 4          BEGIN
: 1092 5              $DS_SETIPL(LEVEL=$X'1F');
: 1093 5              DO
: 1094 5                  BEGIN
: 1095 5                      ! Scope loop starts here
: 1096 6                      UET_CTRL_REG = $X'8000';
: 1097 6                      $DS_SETIPL(LEVEL=.IPL);
: 1098 6                      INTERRUPT_EXP = 1;
: 1099 6                      INTERRUPT_OCC = 0;
: 1100 6                      TEMP = .UET_CTRL_REG2;
: 1101 6                      UET_CTRL_REG2 = (.TEMP AND $X'1FF') OR BR5 * 2;
: 1102 6
: 1103 6                      $DS_WAITUS(TIME=20);
: 1104 6                      IF .INTERRUPT_OCC
: 1105 6                      THEN
: 1106 6                          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: 1107 6                          PRLINK = PRINT_GENERAL,
: 1108 6                          P1 = FMT_UNX_INT1,P2 = 2,
: 1109 6                          P3 = 5,P4 = .IPL);
: 1110 6
: 1111 6      Test 28, Subtest 3, Error 9
: 1112 6      END
: 1113 5      WHILE $DS_INLOOP;          ! Scope loop ends here
: 1114 4      END;
: 1115 3      END;
: 1116 3
: 1117 3      CODECOMMENT
: 1118 3      'Now lower the IPL to 14 and after a nominal wait check to see if the',
: 1119 3      'interrupt has occurred.',
: 1120 3      '':
: 1121 3      BEGIN
: 1122 4          DO
: 1123 4              BEGIN
: 1124 4                  ! Scope loop begins here
: 1125 5                  UET_CTRL_REG = $X'8000';
: 1126 5                  INTERRUPT_EXP = 1;
: 1127 5

```



```

; 1128 5      INTERRUPT_OCC = 0;
; 1129 5      $DS_SETIPL(LEVEL=XX'14');
; 1130 5      TEMP = .UET_CTRL_REG2;
; 1131 5      UET_CTRL_REG2 = (.TEMP AND XX'1FF') OR BR5 * 2;
; 1132 5      $DS_WAITUS(TIME=20);
; 1133 5      IF NOT .INTERRUPT_OCC
; 1134 5      THEN
; P 1135 5          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; P 1136 5              PRLINK = PRINT_GENERAL,
; P 1137 5              P1 = FMT_NO_INT,P2 = 2,
; L 1138 5              P3 = 5,P4 = 14);
; %PRINT:      Test 28, Subtest 3, Error 10
; 1139 5      END
; 1140 4      WHILE $DS_INLOOP;          ! Scope loop ends here
; 1141 4
; 1142 3      END;
; 1143 3
; 1144 2      $DS_ENDSUB;                ! End of Subtest 3
; 1145 2
; 1146 3      $DS_BGNSUB;                ! Subtest 4: BR4
; 1147 3
; 1148 3      CODECOMMENT
; 1149 3      ''
; 1150 3      'Set the IPL to 1F (hex), set the interrupt expected flag, clear the',
; 1151 3      'interrupt occurred flag, and initiate an interrupt at BR4. Lower the IPL',
; 1152 3      'from 1F to 14 by steps of one, checking each time that the interrupt has',
; 1153 3      'not occurred.',
; 1154 3      '';
; 1155 4      BEGIN
; 1156 4
; 1157 4      DECR IPL FROM XX'1F' TO XX'14' DO
; 1158 5      BEGIN
; 1159 5      $DS_SETIPL(LEVEL=XX'1F');
; 1160 5      DO                          ! Scope loop starts here
; 1161 6      BEGIN
; 1162 6      UET_CTRL_REG = XX'8000';
; 1163 6      $DS_SETIPL(LEVEL=.IPL);
; 1164 6      INTERRUPT_EXP = 1;
; 1165 6      INTERRUPT_OCC = 0;
; 1166 6      TEMP = .UET_CTRL_REG2;
; 1167 6      UET_CTRL_REG2 = (.TEMP AND XX'1FF') OR BR4 * 2;
; 1168 6
; 1169 6      $DS_WAITUS(TIME=20);
; 1170 6      IF .INTERRUPT_OCC
; 1171 6      THEN
; P 1172 6          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; P 1173 6              PRLINK = PRINT_GENERAL,
; P 1174 6              P1 = FMT_UNX_INT1,P2 = 2,
; L 1175 6              P3 = 4,P4 = .IPL);
; %PRINT:      Test 28, Subtest 4, Error 11
; 1176 6
; 1177 6      END
; 1178 5      WHILE $DS_INLOOP;          ! Scope loop ends here
; 1179 4      END;
; 1180 3      END;

```

! A8

```

; 1181 3
; 1182 3
; 1183 3
; 1184 3
; 1185 3
; 1186 3
; 1187 4
; 1188 4
; 1189 4
; 1190 5
; 1191 5
; 1192 5
; 1193 5
; 1194 5
; 1195 5
; 1196 5
; 1197 5
; 1198 5
; 1199 5
; P 1200 5
; P 1201 5
; P 1202 5
; L 1203 5
; xPRINT:
; 1204 5
; 1205 4
; 1206 4
; 1207 3
; 1208 3
; 1209 2
; 1210 2
; 1211 3
; 1212 3
; 1213 3
; 1214 3
; 1215 3
; 1216 3
; 1217 3
; 1218 3
; 1219 3
; 1220 4
; 1221 4
; 1222 4
; 1223 4
; 1224 4
; 1225 4
; 1226 4
; 1227 5
; 1228 5
; 1229 5
; 1230 5
; 1231 6
; 1232 6
; 1233 6
; 1234 6

CODECOMMENT
,,
'Now lower the IPL to 13 and after a nominal wait check to see if the',
'interrupt has occurred.',
,,
    BEGIN
        DO                                ! Scope loop begins here
            BEGIN
                UET_CTRL_REG = %X'8000';
                INTERRUPT_EXP = 1;
                INTERRUPT_OCC = 0;
                $DS_SETIPL(LEVEL=%X'13');
                TEMP = .UET_CTRL_REG2;
                UET_CTRL_REG2 = (.TEMP AND %X'1FF') OR BR4 * 2;
                $DS_WAITUS(TIME=20);
                IF NOT .INTERRUPT_OCC
                THEN
                    $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
                                PRLINK = PRINT_GENERAL,
                                P1 = FMT_NO_INT,P2 = 2,
                                P3 = 4,P4 = 13);
            Test 28, Subtest 4, Error 12
            END
            WHILE $DS_INLOOP;                ! Scope loop ends here
        END;
    $DS_ENDSUB;                            ! End of Subtest 4
    $DS_BGNSUB;

CODECOMMENT
,,
'Set the IPL to 13, then float a ONE thru the Interrupt Vector (bits <8:2>)',
'of the UET, forcing an interrupt on level 4. After each Interrupt, check',
'the VECTOR that the interrupt occurred at.',
,,
    BEGIN
        LOCAL
            EXP_VECTOR;
        $DS_SETIPL(LEVEL=%X'13');
        INCR I FROM 2 TO 8 DO
            BEGIN
                EXP_VECTOR = (1 ^ .I);
                UET_CTRL_REG2 = .EXP_VECTOR;
                DO                                ! Scope loop begins here
                    BEGIN
                        UET_CTRL_REG = %X'8000';
                        $DS_WAITUS(TIME=20);
                        INTERRUPT_EXP = 1;

```

```

: 1235 6      TEMP = .UET_CTRL_REG2;
: 1236 6      UET_CTRL_REG2 = (.TEMP AND %X'1FF') OR BR4 * 2;      ! Force the Interrupt
: 1237 6      $DS_WAITUS(TIME=20);
: 1238 6      IF .INT_VECTOR NEQ .EXP_VECTOR
: 1239 6      THEN
: 1240 7          BEGIN
: P 1241 7              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD,
: P 1242 7                  PRLINK = PRINT_GENERAL,
: P 1243 7                  P1 = FMT_VECTOR,P2 = 2,
: L 1244 7                  P3 = .EXP_VECTOR,P4 = .INT_VECTOR);
: %PRINT:      Test 28, Subtest 5, Error 13
: 1245 6          END;
: 1246 6      END
: 1247 5      WHILE $DS_INLOOP;      ! Scope loop ends here
: 1248 4      END;
: 1249 3      END;
: 1250 3
: 1251 3      CODECOMMENT
: 1252 3      ''
: 1253 3      'Set the IPL to 13, then float a ZERO thru the Interrupt Vector (bits <8:2>)',
: 1254 3      'of the UET, forcing an interrupt on level 4. After each Interrupt, check',
: 1255 3      'the VECTOR that the interrupt occurred at.',
: 1256 3      '';
: 1257 4      BEGIN
: 1258 4      LOCAL
: 1259 4          EXP_VECTOR;
: 1260 4
: 1261 4      $DS_SETIPL(LEVEL=%X'13');
: 1262 4
: 1263 4      EXP_VECTOR = %X'1F8';
: 1264 4      INCR I FROM 1 TO 7 DO
: 1265 5          BEGIN
: 1266 5              UET_CTRL_REG2 = .EXP_VECTOR;
: 1267 5              DO      ! Scope loop begins here
: 1268 6                  BEGIN
: 1269 6                      UET_CTRL_REG = %X'8000';
: 1270 6                      $DS_WAITUS(TIME=20);
: 1271 6                      INTERRUPT_EXP = 1;
: 1272 6                      TEMP = .UET_CTRL_REG2;
: 1273 6                      UET_CTRL_REG2 = (.TEMP AND %X'1FF') OR BR4 * 2;      ! Force the Interrupt
: 1274 6                      $DS_WAITUS(TIME=20);
: 1275 6                      IF .INT_VECTOR NEQ .EXP_VECTOR
: 1276 6                      THEN
: 1277 7                          BEGIN
: P 1278 7                              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD,
: P 1279 7                                  PRLINK = PRINT_GENERAL,
: P 1280 7                                  P1 = FMT_VECTOR,P2 = 2,
: L 1281 7                                  P3 = .EXP_VECTOR,P4 = .INT_VECTOR);
: %PRINT:      Test 28, Subtest 5, Error 14
: 1282 6          END;
: 1283 6      END
: 1284 5      WHILE $DS_INLOOP;      ! Scope loop ends here
: 1285 5      EXP_VECTOR = ((.EXP_VECTOR OR 2) ^ 1) AND %X'1FC';
: 1286 4      END;
: 1287 3      END;

```



```
; 1288 3
; 1289 3
; 1290 3
; 1291 3 CODECOMMENT
; 1292 3
; 1293 3 'If there are UBEs present, set their Vectors again.',
; 1294 3
; 1295 4 BEGIN
; 1296 4 IF .NUM_UBE NEQ 0
; 1297 4 THEN
; 1298 4     INCR I FROM 0 TO .NUM_UBE - 1 DO
; 1299 4         $DS_SETVEC(VECTOR=.UBE_VECTOR[I],SRVADR=.UBE_SERVICE[I]);
; 1300 3     END;
; 1301 3
; 1302 2 $DS_ENDSUB;
; 1303 2
; 1304 2 $DS_CHANNEL(UNIT=.LUN,FUNC=CHC$_DSINT);
; 1305 2 ![[1.4] DS_CHANNEL moved outside DS_ENDSUB for AUTO_QA Multiple Subtest Loops.
; 1306 2 ![[1.4] Problem was interrupts being disabled after 1st pass.
; 1307 2
; 1308 1 $DS_ENDTEST;
```

```
.PSECT DISPATCH,NOWRT,NOEXE,2
00000000' 00000000' 00000000' 00000000' 00018 $: .ADDRESS P.AAD, P.AAE, P.AAF, TEST_028
00000000 00000003 00028 .LONG 3, 0
.PSECT $PLIT$,NOWRT,NOEXE,2
72 65 74 6E 49 20 32 52 53 43 20 54 45 55 17 00020 P.AAD: .WORD 28
74 73 65 54 20 74 70 75 72 00022 P.AAE: .ASCII <23>\UET CSR2 Interrupt Test\
00031
0003A .BLKB 2
00000000 0003C P.AAF: .LONG 0
.EXTRN DS$PRINTF
.PSECT TEST_028,NOWRT,2
OFFC 00000
5B 0000G CF 9E 00002 MOVAB INTERRUPT_OCC, R11
5A 0000G CF 9E 00007 MOVAB LUN, R10
59 00000000G 9F 9E 0000C MOVAB @DS$INLOOP, R9
58 00000000G 9F 9E 00013 MOVAB @DS$ERRHARD, R8
57 00000000G 9F 9E 0001A MOVAB @DS$WAITUS, R7
56 00000000G 9F 9E 00021 MOVAB @DS$SETIPL, R6
55 0000G CF 9E 00028 MOVAB UNIBUS_SPACE, R5
50 0000G CF D0 0002D MOVL UBI_LINK, R0
01 0B A0 91 00032 CMPB 11(R0), #1
2C 12 00036 BNEQ 1$
50 65 0003F466 8F C9 00038 BISL3 #259174, UNIBUS_SPACE, R0
52 60 B0 00040 MOVW (R0), TEMP
0844
0848
```

```

                                22 18 00043      BGEQ      2$                ; 0849
                                9F D1 00045      CMPL      a#^X0000FE54, #2    ; 0856
                                16 18 0004C      BGEQ      1$                ;
                                0000G CF 95 0004E    TSTB      EXTERNAL_FLAG    ;
                                10 12 00052      BNEQ      1$                ;
                                0000G CF 9F 00054    PUSHAB   FMT_EXT_PROC    ; 0859
00000000G 9F 01 FB 00058      CALLS      #1, a#DS$PRINTF    ;
0000G CF 01 90 0005F      MOVW      #1, EXTERNAL_FLAG    ; 0860
                                50 D4 00064 1$:      CLRL      R0                ; 0862
                                04 00066      RET                ;
                                01 DD 00067 2$:      PUSHL     #1                ; 0864
                                1C DD 00069      PUSHL     #28              ;
00000000G 9F 02 FB 0006B      CALLS      #2, a#DS$BGNSUB    ;
                                ;
                                ; If there are UBEs selected, release their vectors.
                                ;
                                54 0000G CF D0 00072    MOVL      NUM_UBE, R4        ; 0873
                                16 13 00077      BEQL      5$                ;
                                53 01 CE 00079      MNEGL     #1, I              ; 0875
                                0D 11 0007C      BRB        4$                ;
                                7E 0000GCF 43 3C 0007E 3$:    MOVZWL   UBE_VECTOR[I], -(SP)    ; 0876
EF 00000000G 9F 01 FB 00084      CALLS      #1, a#DS$CLRVEC    ;
                                53 54 F2 0008B 4$:      AOBLS    R4, I, 3$        ;
                                00000000 9F D4 0008F 5$:      CLRL      a#^X00000000    ; 0878
                                7E D4 00095      CLRL      -(SP)              ;
                                0000G CF 9F 00097      PUSHAB   CHAN_STAT        ;
                                0000G CF 9F 0009B      PUSHAB   UET_ISR_2        ;
                                04 DD 0009F      PUSHL     #4                ;
                                7E 6A 9A 000A1      MOVZBL   LUN, -(SP)          ;
00000000G 9F 05 FB 000A4      CALLS      #5, a#DS$CHANNEL    ;
50 65 0003F466 8F C9 000AB      BISL3     #259174, UNIBUS_SPACE, R0    ;
60 0140 8F B0 000B3      MOVW      #320, (R0)                ; 0879
                                ;
                                ; Set the IPL to 1F (hex), set the interrupt expected flag, clear the
                                ; interrupt occurred flag, and initiate an interrupt at BR7. Lower the IPL
                                ; from 1F to 17 by steps of one, checking each time that the interrupt has
                                ; not occurred.
                                ;
                                53 1F D0 000B8      MOVL      #31, IPL                ; 0890
                                1F DD 000BB 6$:      PUSHL     #31                ; 0892
                                66 01 FB 000BD      CALLS      #1, DS$SETIPL    ;
50 65 0003F464 8F C9 000C0 7$:      BISL3     #259172, UNIBUS_SPACE, R0    ; 0894
60 8000 8F B0 000C8      MOVW      #-32768, (R0)            ; 0895
                                53 DD 000CD      PUSHL     IPL                ; 0896
                                66 01 FB 000CF      CALLS      #1, DS$SETIPL    ;
                                0000G CF 01 90 000D2    MOVW      #1, INTERRUPT_EXP    ; 0897
                                6B 94 000D7      CLRB      INTERRUPT_OCC    ; 0898
50 65 0003F466 8F C9 000D9      BISL3     #259174, UNIBUS_SPACE, R0    ; 0899
52 60 B0 000E1      MOVW      (R0), TEMP                ;
51 09 00  EF 000E4      EXTZV     #0, #9, TEMP, R1          ; 0900
60 51 1000 8F A9 000E9      BISW3     #4096, R1, (R0)        ;
7E 14 7D 000EF      MOVQ      #20, -(SP)                ; 0902

```

```

        67      02 FB 000F2      CALLS #2, DS$WAITUS      ;
        1B      6B E9 000F5      BLBC  INTERRUPT_OCC, 8$      ; 0903
;
; Interrupt occurred at IPL greater than or equal to 17
;
        7E      7C 000F8      CLRQ  -(SP)      ; 0913
        53      DD 000FA      PUSHL  IPL      ;
        07      DD 000FC      PUSHL  #7      ;
        02      DD 000FE      PUSHL  #2      ;
        0000G   CF 9F 00100      PUSHAB FMT_UNX_INT1      ;
        0000G   CF 9F 00104      PUSHAB PRINT_GENERAL      ;
        0000G   CF 9F 00108      PUSHAB UBIMOD_CSR      ;
        7E      6A 9A 0010C      MOVZBL LUN, -(SP)      ;
        01      DD 0010F      PUSHL  #1      ;
        30      11 00111      BRB    9$      ;
;
; Check that the interrupt done bit in CSR2 is not set
;
50      65 0003F466      8F C9 00113 8$: BISL3 #259174, UNIBUS_SPACE, R0      ; 0921
        52      60 B0 0011B      MOVW  (R0), TEMP      ;
24      52      0E E1 0011E      BBC   #14, TEMP, 10$      ; 0922
        7E      7C 00122      CLRQ  -(SP)      ; 0927
        50      52 3C 00124      MOVZWL TEMP, R0      ;
        7E      50 FFFFBFFF      8F CB 00127      BICL3 #-16385, R0, -(SP)      ;
        7E      02 7D 0012F      MOVQ  #2, -(SP)      ;
        0000G   CF 9F 00132      PUSHAB FMT_UET_INTD2      ;
        0000G   CF 9F 00136      PUSHAB PRINT_GENERAL      ;
        0000G   CF 9F 0013A      PUSHAB UBIMOD_CSR      ;
        7E      6A 9A 0013E      MOVZBL LUN, -(SP)      ;
        02      DD 00141      PUSHL  #2      ;
        68      0A FB 00143 9$: CALLS #10, DS$ERRHARD      ;
        69      00 FB 00146 10$: CALLS #0, DS$INLOOP      ; 0931
        03      50 E9 00149      BLBC  R0, 11$      ;
        FF65    53      FF 8F      FF71 31 0014C      BRW    7$      ;
        17      9D 0014F 11$: ACBB  #23, #-1, IPL, 6$      ; 0890
;
; Now lower the IPL to 16 and after a nominal wait check to see
; if the
; interrupt has occurred.
;
50      65 0003F464      8F C9 00156 12$: BISL3 #259172, UNIBUS_SPACE, R0      ; 0943
        60      8000      8F B0 0015E      MOVW  #-32768, (R0)      ; 0944
        0000G   CF      01 90 00163      MOVB  #1, INTERRUPT_EXP      ; 0945
        6B      94 00168      CLRB  INTERRUPT_OCC      ; 0946
        16      DD 0016A      PUSHL  #22      ; 0947
        01      FB 0016C      CALLS  #1, DS$SETIPL      ;
50      65 0003F466      8F C9 0016F      BISL3 #259174, UNIBUS_SPACE, R0      ; 0948
        52      60 B0 00177      MOVW  (R0), TEMP      ;
        51      09      00 EF 0017A      EXTZV #0, #9, TEMP, R1      ; 0949
        60      51      1000      8F A9 0017F      BISW3 #4096, R1, (R0)      ;
        7E      14 7D 00185      MOVQ  #20, -(SP)      ; 0950
        67      02 FB 00188      CALLS  #2, DS$WAITUS      ;
        1B      6B E8 0018B      BLBS  INTERRUPT_OCC, 13$      ; 0951
;
; Interrupt did not occur.

```



```

;
; 7E 7C 0018E CLRQ -(SP) ; 0961
; 16 DD 00190 PUSHL #22 ;
; 07 DD 00192 PUSHL #7 ;
; 02 DD 00194 PUSHL #2 ;
; 0000G CF 9F 00196 PUSHAB FMT_NO_INT ;
; 0000G CF 9F 0019A PUSHAB PRINT_GENERAL ;
; 0000G CF 9F 0019E PUSHAB UBIMOD_CSR ;
7E 6A 9A 001A2 MOVZBL LUN, -(SP) ;
; 03 DD 001A5 PUSHL #3 ;
; 6C 11 001A7 BRB 15$ ;
;
; Check that the interrupt done bit in CSR2 is set.
;
50 65 0003F466 8F C9 001A9 13$: BISL3 #259174, UNIBUS_SPACE, R0 ; 0969
; 52 60 B0 001B1 MOVW (R0), TEMP ;
27 52 0E E0 001B4 BBS #14, TEMP, 14$ ; 0970
; 7E 7C 001B8 CLRQ -(SP) ; 0975
; 50 52 3C 001BA MOVZWL TEMP, R0 ;
7E 50 FFFFBFFF 8F CB 001BD BICL3 #-16385, R0, -(SP) ;
; 7E 4000 8F 3C 001C5 MOVZWL #16384, -(SP) ;
; 02 DD 001CA PUSHL #2 ;
; 0000G CF 9F 001CC PUSHAB FMT_UET_INTD2 ;
; 0000G CF 9F 001D0 PUSHAB PRINT_GENERAL ;
; 0000G CF 9F 001D4 PUSHAB UBIMOD_CSR ;
7E 6A 9A 001D8 MOVZBL LUN, -(SP) ;
; 04 DD 001DB PUSHL #4 ;
; 6D 11 001DD BRB 17$ ;
;
; Check that the interrupt done bit clears properly.
; First try and clear it by writting a zero and check
; that it does not clear.
;
50 65 0003F466 8F C9 001DF 14$: BISL3 #259174, UNIBUS_SPACE, R0 ; 0983
; 60 B4 001E7 CLRW (R0) ; 0984
; 52 60 B0 001E9 MOVW (R0), TEMP ; 0985
27 52 0E E0 001EC BBS #14, TEMP, 16$ ; 0986
; 7E 7C 001F0 CLRQ -(SP) ; 0991
; 50 52 3C 001F2 MOVZWL TEMP, R0 ;
7E 50 FFFFBFFF 8F CB 001F5 BICL3 #-16385, R0, -(SP) ;
; 7E 4000 8F 3C 001FD MOVZWL #16384, -(SP) ;
; 02 DD 00202 PUSHL #2 ;
; 0000G CF 9F 00204 PUSHAB FMT_UET_INTD2 ;
; 0000G CF 9F 00208 PUSHAB PRINT_GENERAL ;
; 0000G CF 9F 0020C PUSHAB UBIMOD_CSR ;
7E 6A 9A 00210 MOVZBL LUN, -(SP) ;
; 05 DD 00213 PUSHL #5 ;
; 35 11 00215 15$: BRB 17$ ;
;
; Now write a one to it and check that it clears
;
50 65 0003F466 8F C9 00217 16$: BISL3 #259174, UNIBUS_SPACE, R0 ; 0997
; 60 4000 8F B0 0021F MOVW #16384, (R0) ; 0998
; 52 60 B0 00224 MOVW (R0), TEMP ; 0999
24 52 0E E1 00227 BBC #14, TEMP, 18$ ; 1000

```

		7E	7C	0022B	CLRQ	-(SP)		1005
	50		52	3C	0022D	MOVZWL	TEMP, R0	
7E	50	FFFFBFFF	8F	CB	00230	BICL3	#-16385, R0, -(SP)	
	7E		02	7D	00238	MOVQ	#2, -(SP)	
		0000G	CF	9F	0023B	PUSHAB	FMT_UET_INTD2	
		0000G	CF	9F	0023F	PUSHAB	PRINT_GENERAL	
		0000G	CF	9F	00243	PUSHAB	UBIMOD_CSR	
	7E		6A	9A	00247	MOVZBL	LUN, -(SP)	
			06	DD	0024A	PUSHL	#6	
	68		0A	FB	0024C	17\$:	CALLS	#10, DS\$ERRHARD
	69		00	FB	0024F	18\$:	CALLS	#0, DS\$INLOOP
	03		50	E9	00252	BLBC	R0, 19\$	1010
			FEFE	31	00255	BRW	12\$	
			01	DD	00258	19\$:	PUSHL	#1
			1C	DD	0025A		PUSHL	#28
00000000G	9F		02	FB	0025C		CALLS	#2, a#DS\$ENDSUB
			02	DD	00263		PUSHL	#2
			1C	DD	00265		PUSHL	#28
00000000G	9F		02	FB	00267		CALLS	#2, a#DS\$BGNSUB

; Set the IPL to 1F (hex), set the interrupt expected flag, clear the
; interrupt occurred flag, and initiate an interrupt at BR6. Lower the IPL
; from 1F to 16 by steps of one, checking each time that the interrupt has
; not occurred.

	53		1F	D0	0026E	MOVL	#31, IPL	1027
			1F	DD	00271	20\$:	PUSHL	#31
	66		01	FB	00273		CALLS	#1, DS\$SETIPL
50	65	0003F464	8F	C9	00276	21\$:	BISL3	#259172, UNIBUS_SPACE, R0
	60	8000	8F	B0	0027E		MOVW	#-32768, (R0)
			53	DD	00283		PUSHL	IPL
	66		01	FB	00285		CALLS	#1, DS\$SETIPL
	0000G		01	90	00288		MOVB	#1, INTERRUPT_EXP
			6B	94	0028D		CLRB	INTERRUPT_OCC
50	65	0003F466	8F	C9	0028F		BISL3	#259174, UNIBUS_SPACE, R0
	52		60	B0	00297		MOVW	(R0), TEMP
51	09		00	EF	0029A		EXTZV	#0, #9, TEMP, R1
	51	0800	8F	A9	0029F		BISW3	#2048, R1, (R0)
	7E		14	7D	002A5		MOVQ	#20, -(SP)
	67		02	FB	002A8		CALLS	#2, DS\$WAITUS
	1C		6B	E9	002AB		BLBC	INTERRUPT_OCC, 22\$
			7E	7C	002AE		CLRQ	-(SP)
			53	DD	002B0		PUSHL	IPL
			06	DD	002B2		PUSHL	#6
			02	DD	002B4		PUSHL	#2
		0000G	CF	9F	002B6		PUSHAB	FMT_UNX_INT1
		0000G	CF	9F	002BA		PUSHAB	PRINT_GENERAL
		0000G	CF	9F	002BE		PUSHAB	UBIMOD_CSR
	7E		6A	9A	002C2		MOVZBL	LUN, -(SP)
			07	DD	002C5		PUSHL	#7
	68		0A	FB	002C7		CALLS	#10, DS\$ERRHARD
	69		00	FB	002CA	22\$:	CALLS	#0, DS\$INLOOP

```

FF9A      53      FF      A6      50      E8 002CD      BLBS      R0, 21$      ;
                        8F      16      9D 002D0      ACBB      #22, #-1, IPL, 20$      ; 1027
;
; Now lower the IPL to 15 and after a nominal wait check to see
; if the
; interrupt has occurred.
;
50          65 0003F464 8F C9 002D7 23$: BISL3      #259172, UNIBUS_SPACE, R0      ; 1060
          60      8000 8F B0 002DF      MOVW      #-32768, (R0)      ; 1061
          0000G CF      01 90 002E4      MOVW      #1, INTERRUPT_EXP      ; 1062
          6B 94 002E9      CLRB      INTERRUPT_OCC      ; 1063
          15 DD 002EB      PUSHL      #21      ; 1064
          01 FB 002ED      CALLS      #1, DS$SETIPL      ;
          50          65 0003F466 8F C9 002F0 BISL3      #259174, UNIBUS_SPACE, R0      ; 1065
          52          60 B0 002F8      MOVW      (R0), TEMP      ;
          51          09      00 EF 002FB      EXTZV      #0, #9, TEMP, R1      ; 1066
          60          51      0800 8F A9 00300 BISW3      #2048, R1, (R0)      ;
          7E          14 7D 00306      MOVQ      #20, -(SP)      ; 1067
          67          02 FB 00309      CALLS      #2, DS$WAITUS      ;
          1C          6B E8 0030C      BLBS      INTERRUPT_OCC, 24$      ; 1068
          7E          7C 0030F      CLRQ      -(SP)      ; 1073
          0F DD 00311      PUSHL      #15      ;
          06 DD 00313      PUSHL      #6      ;
          02 DD 00315      PUSHL      #2      ;
          0000G CF 9F 00317      PUSHAB      FMT_NO_INT      ;
          0000G CF 9F 0031B      PUSHAB      PRINT_GENERAL      ;
          0000G CF 9F 0031F      PUSHAB      UBIMOD_CSR      ;
          7E          6A 9A 00323      MOVZBL      LUN, -(SP)      ;
          08 DD 00326      PUSHL      #8      ;
          68          0A FB 00328      CALLS      #10, DS$ERRHARD      ;
          69          00 FB 0032B 24$: CALLS      #0, DS$INLOOP      ; 1075
          A6          50 E8 0032E      BLBS      R0, 23$      ;
          02 DD 00331      PUSHL      #2      ; 1077
          1C DD 00333      PUSHL      #28      ;
          00000000G 9F 02 FB 00335      CALLS      #2, a#DS$ENDSUB      ;
          03 DD 0033C      PUSHL      #3      ; 1079
          1C DD 0033E      PUSHL      #28      ;
          00000000G 9F 02 FB 00340      CALLS      #2, a#DS$BGNSUB      ;
;
; Set the IPL to 1F (hex), set the interrupt expected flag, cle
; ar the
; interrupt occurred flag, and initiate an interrupt at BR5. L
; ower the IPL
; from 1F to 15 by steps of one, checking each time that the in
; terrupt has
; not occurred.
;
          53          1F D0 00347      MOVL      #31, IPL      ; 1092
          1F DD 0034A 25$: PUSHL      #31      ; 1094
          66          01 FB 0034C      CALLS      #1, DS$SETIPL      ;
          50          65 0003F464 8F C9 0034F 26$: BISL3      #259172, UNIBUS_SPACE, R0      ; 1096
          60          8000 8F B0 00357      MOVW      #-32768, (R0)      ; 1097
          53          DD 0035C      PUSHL      IPL      ; 1098
          01 FB 0035E      CALLS      #1, DS$SETIPL      ;
          0000G CF 01 90 00361      MOVW      #1, INTERRUPT_EXP      ; 1099

```


			6B	94	00366	CLRB	INTERRUPT_OCC	:	1100
	50	65	8F	C9	00368	BISL3	#259174, UNIBUS_SPACE, R0	:	1101
		52	60	B0	00370	MOVW	(R0), TEMP	:	
51	52	09	00	EF	00373	EXTZV	#0, #9, TEMP, R1	:	1102
	60	51	8F	A9	00378	BISW3	#1024, R1, (R0)	:	
		7E	14	7D	0037E	MOVQ	#20, -(SP)	:	1104
		67	02	FB	00381	CALLS	#2, DS\$WAITUS	:	
		1C	6B	E9	00384	BLBC	INTERRUPT_OCC, 27\$	:	1105
			7E	7C	00387	CLRQ	-(SP)	:	1110
			53	DD	00389	PUSHL	IPL	:	
			05	DD	0038B	PUSHL	#5	:	
			02	DD	0038D	PUSHL	#2	:	
		0000G	CF	9F	0038F	PUSHAB	FMT_UNX_INT1	:	
		0000G	CF	9F	00393	PUSHAB	PRINT_GENERAL	:	
		0000G	CF	9F	00397	PUSHAB	UBIMOD_CSR	:	
		7E	6A	9A	0039B	MOVZBL	LUN, -(SP)	:	
			09	DD	0039E	PUSHL	#9	:	
		68	0A	FB	003A0	CALLS	#10, DS\$ERRHARD	:	
		69	00	FB	003A3	CALLS	#0, DS\$INLOOP	:	1113
		A6	50	E8	003A6	BLBS	R0, 26\$	:	
FF9A	53	FF	15	9D	003A9	ACBB	#21, #-1, IPL, 25\$	:	1092

; Now lower the IPL to 14 and after a nominal wait check to see
if the
interrupt has occurred.

	50	65	8F	C9	003B0	28\$:	BISL3	#259172, UNIBUS_SPACE, R0	:	1125
		60	8F	B0	003B8		MOVW	#-32768, (R0)	:	1126
		CF	01	90	003BD		MOVB	#1, INTERRUPT_EXP	:	1127
			6B	94	003C2		CLRB	INTERRUPT_OCC	:	1128
			14	DD	003C4		PUSHL	#20	:	1129
		66	01	FB	003C6		CALLS	#1, DS\$SETIPL	:	
	50	65	8F	C9	003C9		BISL3	#259174, UNIBUS_SPACE, R0	:	1130
		52	60	B0	003D1		MOVW	(R0), TEMP	:	
51	52	09	00	EF	003D4		EXTZV	#0, #9, TEMP, R1	:	1131
	60	51	8F	A9	003D9		BISW3	#1024, R1, (R0)	:	
		7E	14	7D	003DF		MOVQ	#20, -(SP)	:	1132
		67	02	FB	003E2		CALLS	#2, DS\$WAITUS	:	
		1C	6B	E8	003E5		BLBS	INTERRUPT_OCC, 29\$	:	1133
			7E	7C	003E8		CLRQ	-(SP)	:	1138
			0E	DD	003EA		PUSHL	#14	:	
			05	DD	003EC		PUSHL	#5	:	
			02	DD	003EE		PUSHL	#2	:	
		0000G	CF	9F	003F0		PUSHAB	FMT_NO_INT	:	
		0000G	CF	9F	003F4		PUSHAB	PRINT_GENERAL	:	
		0000G	CF	9F	003F8		PUSHAB	UBIMOD_CSR	:	
		7E	6A	9A	003FC		MOVZBL	LUN, -(SP)	:	
			0A	DD	003FF		PUSHL	#10	:	
		68	0A	FB	00401		CALLS	#10, DS\$ERRHARD	:	
		69	00	FB	00404	29\$:	CALLS	#0, DS\$INLOOP	:	1140
		A6	50	E8	00407		BLBS	R0, 28\$	:	
			03	DD	0040A		PUSHL	#3	:	1142
			1C	DD	0040C		PUSHL	#28	:	
		00000000G	9F	02	FB	0040E	CALLS	#2, a#DS\$ENDSUB	:	
			04	DD	00415		PUSHL	#4	:	1144

```

00000000G 9F      1C DD 00417      PUSHL #28      ;
02 FB 00419      CALLS #2, a#DS$BGNSUB      ;
;
; Set the IPL to 1F (hex), set the interrupt expected flag, cle
; ar the
; interrupt occurred flag, and initiate an interrupt at BR4. L
; ower the IPL
; from 1F to 14 by steps of one, checking each time that the in
; terrupt has
; not occurred.
;
53      1F D0 00420      MOVL #31, IPL      ; 1157
      1F DD 00423 30$: PUSHL #31      ; 1159
      01 FB 00425      CALLS #1, DS$SETIPL      ;
50      66      0003F464 8F C9 00428 31$: BISL3 #259172, UNIBUS_SPACE, R0      ; 1161
      60      8000      8F B0 00430      MOVW #-32768, (R0)      ; 1162
      53 DD 00435      PUSHL IPL      ; 1163
      66      01 FB 00437      CALLS #1, DS$SETIPL      ;
      0000G CF      01 90 0043A      MOVW #1, INTERRUPT_EXP      ; 1164
      6B 94 0043F      CLRB INTERRUPT_OCC      ; 1165
50      65 0003F466 8F C9 00441      BISL3 #259174, UNIBUS_SPACE, R0      ; 1166
      52      60 B0 00449      MOVW (R0), TEMP      ;
51      52      09      00 EF 0044C      EXTZV #0, #9, TEMP, R1      ; 1167
      60      51      0200 8F A9 00451      BISW3 #512, R1, (R0)      ;
      7E      14 7D 00457      MOVQ #20, -(SP)      ; 1169
      67      02 FB 0045A      CALLS #2, DS$WAITUS      ;
      1C      6B E9 0045D      BLBC INTERRUPT_OCC, 32$      ; 1170
      7E 7C 00460      CLRQ -(SP)      ; 1175
      53 DD 00462      PUSHL IPL      ;
      04 DD 00464      PUSHL #4      ;
      02 DD 00466      PUSHL #2      ;
      0000G CF 9F 00468      PUSHAB FMT_UNX_INT1      ;
      0000G CF 9F 0046C      PUSHAB PRINT_GENERAL      ;
      0000G CF 9F 00470      PUSHAB UBIMOD_CSR      ;
      7E      6A 9A 00474      MOVZBL LUN, -(SP)      ;
      0B DD 00477      PUSHL #11      ;
      68      0A FB 00479      CALLS #10, DS$ERRHARD      ;
      69      00 FB 0047C 32$: CALLS #0, DS$INLOOP      ; 1178
      A6      50 E8 0047F      BLBS R0, 31$      ;
      FF9A      53      FF 8F      14 9D 00482      ACBB #20, #-1, IPL, 30$      ; 1157
;
; Now lower the IPL to 13 and after a nominal wait check to see
; if the
; interrupt has occurred.
;
50      65 0003F464 8F C9 00489 33$: BISL3 #259172, UNIBUS_SPACE, R0      ; 1190
      60      8000      8F B0 00491      MOVW #-32768, (R0)      ; 1191
      0000G CF      01 90 00496      MOVW #1, INTERRUPT_EXP      ; 1192
      6B 94 0049B      CLRB INTERRUPT_OCC      ; 1193
      13 DD 0049D      PUSHL #19      ; 1194
      66      01 FB 0049F      CALLS #1, DS$SETIPL      ;
50      65 0003F466 8F C9 004A2      BISL3 #259174, UNIBUS_SPACE, R0      ; 1195
      52      60 B0 004AA      MOVW (R0), TEMP      ;
51      52      09      00 EF 004AD      EXTZV #0, #9, TEMP, R1      ; 1196
      60      51      0200 8F A9 004B2      BISW3 #512, R1, (R0)      ;

```

7E		14	7D	004B8	MOVQ	#20, -(SP)	; 1197
67		02	FB	004BB	CALLS	#2, DS\$WAITUS	; 1198
1C		6B	E8	004BE	BLBS	INTERRUPT_OCC, 34\$	; 1203
		7E	7C	004C1	CLRQ	-(SP)	; 1205
		0D	DD	004C3	PUSHL	#13	; 1207
		04	DD	004C5	PUSHL	#4	; 1209
		02	DD	004C7	PUSHL	#2	; 1224
	0000G	CF	9F	004C9	PUSHAB	FMT_NO_INT	; 1226
	0000G	CF	9F	004CD	PUSHAB	PRINT_GENERAL	; 1228
	0000G	CF	9F	004D1	PUSHAB	UBIMOD_CSR	; 1231
7E		6A	9A	004D5	MOVZBL	LUN, -(SP)	; 1233
		0C	DD	004D8	PUSHL	#12	; 1234
68		0A	FB	004DA	CALLS	#10, DS\$ERRHARD	; 1235
69		00	FB	004DD	CALLS	#0, DS\$INLOOP	; 1236
A6		50	E8	004E0	BLBS	R0, 33\$	; 1237
		04	DD	004E3	PUSHL	#4	; 1238
		1C	DD	004E5	PUSHL	#28	; 1244
00000000G	9F	02	FB	004E7	CALLS	#2, a#DS\$ENDSUB	; 1244
		05	DD	004EE	PUSHL	#5	; 1244
		1C	DD	004F0	PUSHL	#28	; 1244
00000000G	9F	02	FB	004F2	CALLS	#2, a#DS\$BGNSUB	; 1244

; Set the IPL to 13, then float a ONE thru the Interrupt Vector
(bits <8:2>)
; of the UET, forcing an interrupt on level 4. After each Inter
rupt, check
; the VECTOR that the interrupt occurred at.

		13	DD	004F9	PUSHL	#19	; 1224
		01	FB	004FB	CALLS	#1, DS\$SETIPL	; 1226
		02	D0	004FE	MOVL	#2, I	; 1228
54		53	78	00501	ASHL	I, #1, EXP_VECTOR	; 1229
50		8F	C9	00505	BISL3	#259174, UNIBUS_SPACE, R0	; 1231
		54	B0	0050D	MOVW	EXP_VECTOR, (R0)	; 1232
50		8F	C9	00510	BISL3	#259172, UNIBUS_SPACE, R0	; 1233
		8F	B0	00518	MOVW	#-32768, (R0)	; 1234
		14	7D	0051D	MOVQ	#20, -(SP)	; 1235
		02	FB	00520	CALLS	#2, DS\$WAITUS	; 1236
	0000G	01	90	00523	MOVB	#1, INTERRUPT_EXP	; 1237
50		8F	C9	00528	BISL3	#259174, UNIBUS_SPACE, R0	; 1238
		60	B0	00530	MOVW	(R0), TEMP	; 1239
		00	EF	00533	EXTZV	#0, #9, TEMP, R1	; 1240
51		8F	A9	00538	BISW3	#512, R1, (R0)	; 1241
		14	7D	0053E	MOVQ	#20, -(SP)	; 1242
		02	FB	00541	CALLS	#2, DS\$WAITUS	; 1243
		CF	D1	00544	CMPL	INT_VECTOR, EXP_VECTOR	; 1244
		1E	13	00549	BEQL	37\$	; 1244
		7E	7C	0054B	CLRQ	-(SP)	; 1244
	0000G	CF	DD	0054D	PUSHL	INT_VECTOR	; 1244
		54	DD	00551	PUSHL	EXP_VECTOR	; 1244
		02	DD	00553	PUSHL	#2	; 1244
	0000G	CF	9F	00555	PUSHAB	FMT_VECTOR	; 1244
	0000G	CF	9F	00559	PUSHAB	PRINT_GENERAL	; 1244
	0000G	CF	9F	0055D	PUSHAB	UBIMOD	; 1244
7E		6A	9A	00561	MOVZBL	LUN, -(SP)	; 1244

	68		0D DD 00564	PUSHL #13	:	
	69		0A FB 00566	CALLS #10, DS\$ERRHARD	:	
	A1		00 FB 00569 37\$:	CALLS #0, DS\$INLOOP	:	1247
	53		50 E8 0056C	BLBS R0, 36\$	:	
8E			08 F3 0056F	AOBLEQ #8, I, 35\$	:	1226
				; Set the IPL to 13, then float a ZERO thru the Interrupt Vector (bits <8:2>)		
				; of the UET, forcing an interrupt on level 4. After each Interrupt, check		
				; the VECTOR that the interrupt occurred at.		
				;		
			13 DD 00573	PUSHL #19	:	1261
	66		01 FB 00575	CALLS #1, DS\$SETIPL	:	
	53	01F8	8F 3C 00578	MOVZWL #504, EXP_VECTOR	:	1263
	54		01 D0 0057D	MOVL #1, I	:	1264
50	65	0003F466	8F C9 00580 38\$:	BISL3 #259174, UNIBUS_SPACE, R0	:	1265
	60		53 B0 00588	MOVW EXP_VECTOR, (R0)	:	1266
50	65	0003F464	8F C9 0058B 39\$:	BISL3 #259172, UNIBUS_SPACE, R0	:	1268
	60	8000	8F B0 00593	MOVW #-32768, (R0)	:	1269
	7E		14 7D 00598	MOVQ #20, -(SP)	:	1270
	67		02 FB 0059B	CALLS #2, DS\$WAITUS	:	
	CF	0000G	01 90 0059E	MOVB #1, INTERRUPT_EXP	:	1271
50	65	0003F466	8F C9 005A3	BISL3 #259174, UNIBUS_SPACE, R0	:	1272
	52		60 B0 005AB	MOVW (R0), TEMP	:	
52	09		00 EF 005AE	EXTZV #0, #9, TEMP, R1	:	1273
60	51	0200	8F A9 005B3	BISW3 #512, R1, (R0)	:	
	7E		14 7D 005B9	MOVQ #20, -(SP)	:	1274
	67		02 FB 005BC	CALLS #2, DS\$WAITUS	:	
	53	0000G	CF D1 005BF	CML INT_VECTOR, EXP_VECTOR	:	1275
			1E 13 005C4	BEQL 40\$	:	
			7E 7C 005C6	CLRQ -(SP)	:	1281
		0000G	CF DD 005C8	PUSHL INT_VECTOR	:	
			53 DD 005CC	PUSHL EXP_VECTOR	:	
			02 DD 005CE	PUSHL #2	:	
		0000G	CF 9F 005D0	PUSHAB FMT_VECTOR	:	
		0000G	CF 9F 005D4	PUSHAB PRINT_GENERAL	:	
		0000G	CF 9F 005D8	PUSHAB UBIMOD	:	
	7E		6A 9A 005DC	MOVZBL LUN, -(SP)	:	
			0E DD 005DF	PUSHL #14	:	
	68		0A FB 005E1	CALLS #10, DS\$ERRHARD	:	
	69		00 FB 005E4 40\$:	CALLS #0, DS\$INLOOP	:	1284
	A1		50 E8 005E7	BLBS R0, 39\$	:	
50	53		01 78 005EA	ASHL #1, EXP_VECTOR, R0	:	1285
	50	FFFFFFE03	8F CA 005EE	BICL2 #-509, R0	:	
53	50		04 C9 005F5	BISL3 #4, R0, EXP_VECTOR	:	
83	54		07 F3 005F9	AOBLEQ #7, I, 38\$	:	1264
				; If there are UBEs present, set their Vectors again.		
				;		
	53	0000G	CF D0 005FD	MOVL NUM_UBE, R3	:	1296
			1D 13 00602	BEQL 43\$	:	
	52		01 CE 00604	MNEGL #1, I	:	1298
			14 11 00607	BRB 42\$	:	
			7E D4 00609 41\$:	CLRL -(SP)	:	1299

ZZ-ECCBA-1.4
UBI_TESTS_27_32
01-04

TEST_028: UET CSR2 Interrupt Test
TEST_028: UET CSR2 Interrupt Test

C 6

6-Sep-1985

Fiche 3 Frame C6

Sequence 479

6-Sep-1985 17:25:18

VAX-11 Bliss-32 V4.1-746

Page 48

6-Sep-1985 17:15:07

[LEMIEUX.ECCBA.RELEASE]ECCBA8.B32;19

(4)

		0000GCF42	DD 0060B	PUSHL	UBE_SERVICE[I]	;
		0000GCF42	3C 00610	MOVZWL	UBE_VECTOR[I], -(SP)	;
E8	00000000G	7E	03 FB 00616	CALLS	#3, a#DS\$SETVEC	;
		9F	53 F2 0061D	AOBLSS	R3, I, 41\$	;
		52	05 DD 00621	PUSHL	#5	;
			1C DD 00623	PUSHL	#28	;
	00000000G	9F	02 FB 00625	CALLS	#2, a#DS\$ENDSUB	;
			9F D4 0062C	CLRL	a#X00000000	;
			7E 7C 00632	CLRQ	-(SP)	;
		7E	05 7D 00634	MOVQ	#5, -(SP)	;
		7E	6A 9A 00637	MOVZBL	LUN, -(SP)	;
	00000000G	9F	05 FB 0063A	CALLS	#5, a#DS\$CHANNEL	;
	00000000G	9F	00 FB 00641	CALLS	#0, a#DS\$BREAK	;
		50	01 D0 00648	MOVL	#1, R0	;
			04 0064B	RET		;

; Routine Size: 1612 bytes, Routine Base: TEST_028 + 0000

; 1309 1

```
; 1310 2 $DS_BGNTTEST (SECTION = (DEFAULT,ALL), TEXT = 'UET CSR1/CSR2 ACLOW Test');
; 1311 2
; 1312 2 !**
; 1313 2 ! [1.4] The documentation for this test was corrected and modified.
; 1314 2 !
; 1315 2 ! TEST DESCRIPTION
; 1316 2 !
; 1317 2 ! This test is only executed for the second unibus and only
; 1318 2 ! if it is not in external processor mode. The test verifies the
; 1319 2 ! functionality of CR1 bit 13 and CR2 bit 13 (ACLO1 and ACLO2).
; 1320 2 !
; 1321 2 ! SUBTEST 1
; 1322 2 !
; 1323 2 ! Verify that the ACLO interrupt does not occur at IPL levels
; 1324 2 ! of 17(X) or higher and that CR1 bit 13 (ACLO1) ascerts when
; 1325 2 ! CR2 bit 13 (ACLO2) is ascerted.
; 1326 2 !
; 1327 2 ! Then, lower the IPL level to 16(X) and verify that the interrupt
; 1328 2 ! occurs.
; 1329 2 !
; 1330 2 ! SUBTEST 2
; 1331 2 !
; 1332 2 ! Verify that an IO RESET (via IPEC INIT) causes CR1 bit 12 (ACIE)
; 1333 2 ! to clear,
; 1334 2 !
; 1335 2 ! SUBTEST 3
; 1336 2 !
; 1337 2 ! Verify that CR1 bit 12 (ACIE) clears with INIT (CR1 bit 15).
; 1338 2 !
; 1339 2 ! SUBTEST 4
; 1340 2 !
; 1341 2 ! Verify that that ACLO interrupt does not occur if the interrupt
; 1342 2 ! enable ACIE CR1 bit 12 is not ascerted.
; 1343 2 !
; 1344 2 ! --
; 1345 2
; 1346 2 REGISTER
; 1347 2 IPL,
; 1348 2 TEMP: WORD,
; 1349 2 I;
; 1350 2
; 1351 2 IF .UBI_LINK[HP$B_DRIVE] NEQ 1
; 1352 2 THEN $DS_ABORT(ARG = TEST)
; 1353 2 ELSE
; 1354 3 BEGIN
; 1355 3 TEMP = .UET_CTRL_REG2;
; 1356 3 IF (.TEMP AND %X'8000') EQL %X'8000'
; 1357 3 THEN
; 1358 4 BEGIN
; 1359 5 IF ((.DSA$GL_PASSNO LSS 2) AND (.EXTERNAL_FLAG EQL 0))
; 1360 4 THEN
; 1361 5 BEGIN
; 1362 5 $DS_PRINTF(FMT_EXT_PROC);
; 1363 5 EXTERNAL_FLAG = 1;
; 1364 4 END;
```



```

; 1365 4      $DS_ABORT(ARG = TEST);
; 1366 3      END;
; 1367 2      END;
; 1368 3      $DS_BGNSUB;                      ! Subtest 1
; 1369 3
; 1370 3      $DS_SETVEC(VECTOR = XX'1E4',SRVADR = UET_ISR_2);
; 1371 3
; 1372 3      CODECOMMENT
; 1373 3      ''
; 1374 3      'Set the IPL to 1F (hex), set the interrupt expected flag, clear the',
; 1375 3      'interrupt occurred flag, and initiate an ACLO2 interrupt at BR7.Lower the IPL',
; 1376 3      'from 1F to 17 by steps of one, checking each time that the interrupt has',
; 1377 3      'not occurred.',
; 1378 3      '';
; 1379 4      BEGIN
; 1380 4
; 1381 4      DECR IPL FROM XX'1F' TO XX'17' DO
; 1382 5      BEGIN
; 1383 5      $DS_SETIPL(LEVEL=XX'1F');
; 1384 5      DO                                ! Scope loop starts here
; 1385 6      BEGIN
; 1386 6      UET_CTRL_REG = XX'8000';          ! Set INIT CR1 <15:00> ,initializes
; 1387 6      $DS_WAITUS(TIME=20);             ! IPEC internal logic.
; 1388 6      UET_CTRL_REG = XX'1000';          ! Set ACIE CR1 <12:00>, enables
; 1389 6      INTERRUPT_EXP = 1;                ! ACLO interrupts
; 1390 6      INTERRUPT_OCC = 0;
; 1391 6      UET_CTRL_REG2 = XX'2000';         ! Set ACLO2 CR2 <13:00>
; 1392 6
; 1393 6      $DS_SETIPL(LEVEL=.IPL);
; 1394 6      $DS_WAITUS(TIME=20);             ! Wait for bits to set
; 1395 6      IF .INTERRUPT_OCC
; 1396 6      THEN
; P 1397 6      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; P 1398 6      PRLINK = PRINT_GENERAL,
; P 1399 6      P1 = FMT_UNX_ACINT,P2=2,
; L 1400 6      P3 = XX'17',P4 = .IPL)
; xPRINT:    Test 29, Subtest 1, Error 1
; 1401 6      ELSE
; 1402 6      CODECOMMENT
; 1403 6      ''
; 1404 6      'Check that the ACLO1 bit is set in CR1.',
; 1405 6      '';
; 1406 7      BEGIN
; 1407 7      TEMP = .UET_CTRL_REG;
; 1408 7      IF (.TEMP AND XX'2000') NEQ XX'2000'
; 1409 7      THEN
; P 1410 7      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; P 1411 7      PRLINK=PRINT_GENERAL,
; P 1412 7      P1=FMT_UET_ACLO1,P2=2,
; L 1413 7      P3=XX'2000',P4=.TEMP AND XX'2000');
; xPRINT:    Test 29, Subtest 1, Error 2
; 1414 6      END;
; 1415 6      UET_CTRL_REG = XX'8000';          ! [1.4] $DS_INLOOP executes at lower IPL
; 1416 6      $DS_WAITMS(TIME=20);              ! [1.4] so ACIE must be cleared.
; 1417 6      $DS_WAITMS(TIME=20);              ! [1.4] Wait for ACLO2 to clear (self clearing)

```

```

; 1418 6      END
; 1419 5      WHILE $DS_INLOOP;          ! Scope loop ends here
; 1420 4      END;
; 1421 3      END;
; 1422 3      CODECOMMENT
; 1423 3      '
; 1424 3      'Now lower the IPL to 16 and after a nominal wait check to see if the',
; 1425 3      'interrupt has occurred. Also check that the ',
; 1426 3      'ACLO1 bit set in CR1.',
; 1427 3      '':
; 1428 4      BEGIN
; 1429 4
; 1430 4      DO                          ! Scope loop begins here
; 1431 5          BEGIN
; 1432 5          UET_CTRL_REG = %X'8000';          ! Clear ACIE using INIT
; 1433 5          $DS_SETIPL(LEVEL=%X'1F');
; 1434 5          UET_CTRL_REG2 = 0;                ! Clear all clearable bits in CR2
; 1435 5          $DS_WAITMS(TIME=20);              ! Wait for ACLO2 to clear (self clearing)
; 1436 5          INTERRUPT_EXP = 1;
; 1437 5          INTERRUPT_OCC = 0;
; 1438 5          $DS_SETIPL(LEVEL=%X'16');
; 1439 5          UET_CTRL_REG = %X'1000';          ! Set interrupt enable CR1<12:0> ACIE
; 1440 5          UET_CTRL_REG2 = %X'2000';        ! Set ACLO2 CR2 <13:00>
; 1441 5          $DS_WAITUS(TIME=20);
; 1442 5          IF NOT .INTERRUPT_OCC
; 1443 5          THEN
; 1444 6              BEGIN
; 1445 6                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; 1446 6                      PRLINK = PRINT_GENERAL,
; 1447 6                      P1 = FMT_NO_ACINT,P2 = 2,
; 1448 6                      P3 = 7,P4 = %X'16');
; 1449 6          Test 29, Subtest 1, Error 3
; 1450 5          ELSE
; 1451 5              CODECOMMENT
; 1452 5              '
; 1453 5              'Check that ACLO1 in CR1 <13:00> is set.',
; 1454 5              '':
; 1455 6              BEGIN
; 1456 6              TEMP = .UET_CTRL_REG;
; 1457 6              IF (.TEMP AND %X'2000') NEQ %X'2000'
; 1458 6              THEN
; 1459 6                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; 1460 6                      PRLINK=PRINT_GENERAL,
; 1461 6                      P1=FMT_UET_ACLO1,P2=2,
; 1462 6                      P3=%X'2000',P4=.TEMP AND %X'2000');
; 1463 5          Test 29, Subtest 1, Error 4
; 1464 5          END;
; 1465 5          UET_CTRL_REG2 = %X'8000';          ! [1.4] Clear ACIE because
; 1466 5          $DS_WAITUS(TIME=20);              ! [1.4] $DS_INLOOP executes at lower IPL
; 1467 4          END
; 1468 4          WHILE $DS_INLOOP;          ! Scope loop ends here
; 1469 4          UET_CTRL_REG = 0;                ! Clear all clearable bits in CR1
; 1470 4          UET_CTRL_REG2 = 0;                ! Clear all clearable bits in CR2
; 1471 4          $DS_WAITMS(TIME=20);              ! Wait for ACLO2 to clear (self clearing)

```

```

: 1471 4      UET_CTRL_REG = XX'8000';      ! Init UET
: 1472 4      $DS_SETIPL(LEVEL=0);
: 1473 3      END;
: 1474 2      $DS_ENDSUB;
: 1475 2
: 1476 3      $DS_BGNSUB;
: 1477 3      CODECOMMENT
: 1478 3      ''
: 1479 3      ' Check that INIT clears the interrupt enable bit in CR1 <12:00> ACIE',
: 1480 3      '';
: 1481 4      BEGIN
: 1482 4      DO                                ! Scope loop starts here
: 1483 5      BEGIN
: 1484 5      UET_CTRL_REG = XX'8000';          ! Set INIT to clear ACIE (CR1 <12:00>)
: 1485 5      $DS_WAITUS(TIME=20);
: 1486 5      UET_CTRL_REG = XX'1000';          ! Set ACIE (CR1 <12:00>)
: 1487 5      $DS_WAITUS(TIME=20);          ! [1.4] Wait for it to set.
: 1488 5      UET_CTRL_REG = XX'8000';          ! Set INIT to clear ACIE (CR1 <12:00>)
: 1489 5      $DS_WAITUS(TIME=20);
: 1490 5      TEMP = UET_CTRL_REG;
: 1491 5      IF (.TEMP AND XX'1000') NEQ 0    ! Find out if it cleared.
: 1492 5      THEN
: 1493 5          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: 1494 5              PRLINK=PRINT_GENERAL,
: 1495 5              P1=FMT_INIT_ACLO,P2=2,
: 1496 5              P3=0,P4=.TEMP AND XX'1000');
: 1497 5      $PRINT: Test 29, Subtest 2, Error 5
: 1498 4      END
: 1499 3      WHILE $DS_INLOOP;
: 1500 2      END;
: 1501 2      $DS_ENDSUB;
: 1502 3
: 1503 3      $DS_BGNSUB;
: 1504 3      CODECOMMENT
: 1505 3      ''
: 1506 3      ' Check that the interrupt is inhibited by the interrupt enable bit.',
: 1507 3      '';
: 1508 4      BEGIN
: 1509 4      DO                                ! Scope loop starts here
: 1510 5      BEGIN
: 1511 5      UET_CTRL_REG = XX'8000';          ! Set INIT (CR1 <15:00>), initializes IPEC
: 1512 5      $DS_WAITUS(TIME=20);          ! clearing ACIE (CR1 <12:00>).
: 1513 5      INTERRUPT_EXP = 1;
: 1514 5      INTERRUPT_OCC = 0;
: 1515 5      UET_CTRL_REG2 = XX'2000';        ! Set ACLO2 (CR2 <13:00>)
: 1516 5      $DS_WAITUS(TIME = 20);
: 1517 5      IF .INTERRUPT_OCC              ! Interrupt should not occur.
: 1518 5      THEN
: 1519 5          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
: 1520 5              PRLINK = PRINT_GENERAL,
: 1521 5              P1 = FMT_UNX_ACINT1,P2=0);
: 1522 5      $PRINT: Test 29, Subtest 3, Error 6
: 1523 5      UET_CTRL_REG2 = 0;              ! Clear all clearable bits in CR2
: 1524 5      $DS_WAITMS(TIME=20);          ! Wait for ACLO2 to clear (self clearing).
: 1525 5      END

```



```

; 1524 4      WHILE $DS_INLOOP;
; 1525 3      END;
; 1526 2      $DS_ENDSUB;
; 1527 2
; 1528 3      $DS_BGNSUB;
; 1529 3      CODECOMMENT
; 1530 3      '
; 1531 3      ' Now check that the interrupt occurs when ACIE and ACLO2 both set',
; 1532 3      '
; 1533 4      BEGIN
; 1534 4      DO                                ! Scope loop starts here
; 1535 5          BEGIN
; 1536 5          UET_CTRL_REG = %X'8000'; ! Set INIT (CR1 <15:00>, initialize IPEC.
; 1537 5          $DS_WAITUS(TIME=20); ! clear ACIE.
; 1538 5          UET_CTRL_REG = %X'1000'; ! Set ACIE (CR1 <12:00>) ,interrupt enable.
; 1539 5          INTERRUPT_EXP = 1;
; 1540 5          UET_CTRL_REG2 = %X'2000'; ! Set ACLO2 (CR2 <13:00>).
; 1541 5          $DS_WAITUS(TIME = 20);
; 1542 5          INTERRUPT_EXP = 1;
; 1543 5          INTERRUPT_OCC = 0;
; 1544 5          UET_CTRL_REG2 = 0; ! Clear all clearable bits of CR2.
; 1545 5          $DS_WAITMS(TIME = 20); ! [1.4] Wait for ACLO2 to clear (self
; 1546 5          ! [1.4] clearing). Was 100 ms ,now 20 ms.
; 1547 5          IF NOT .INTERRUPT_OCC ! Interrupt should occur.
; 1548 5          THEN
; P 1549 5              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_CSR,
; P 1550 5                  PRLINK=PRINT_GENERAL,
; L 1551 5                  P1 = FMT_NO_ACINT1,P2 = 0);
; *PRINT:      Test 29, Subtest 4, Error 7
; 1552 5      END
; 1553 4      WHILE $DS_INLOOP;
; 1554 3      END;
; 1555 3
; 1556 3
; 1557 3
; 1558 2      $DS_ENDSUB;
; 1559 2
; 1560 2      $DS_CLRVEC(VECTOR=%X'1E4');
; 1561 2      ![[1.4] DS_CLRVEC moved outside DS_ENDSUB for AUTO_QA Multiple Subtest Loops.
; 1562 2      ![[1.4] Problem was interrupt had no vector after first pass.
; 1563 2
; 1564 1      $DS_ENDTEST;

```

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00030 \$:
00000000 00000003 00040

.ADDRESS P.AAG, P.AAH, P.AAI, TEST_029
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

20	32	52	53	43	2F	31	52	53	43	20	54	45	55	18	00040	P.AAG:	.WORD	29	:
					74	73	65	54	20	57	4F	4C	43	41	00042	P.AAH:	.ASCII	<24>\UET CSR1/CSR2 ACLOW Test\	:
															00051				:

00000000 0005B .BLKB 1
0005C P.AAI: .LONG 0

.EXTRN DS\$WAITMS

.PSECT TEST_029,NOWRT,2

OFFC 00000

.ENTRY TEST_029, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1310
R11 ;

5B 0000G CF 9E 00002
5A 00000000G 9F 9E 00007
59 00000000G 9F 9E 0000E
58 00000000G 9F 9E 00015
57 00000000G 9F 9E 0001C
56 00000000G 9F 9E 00023
55 00000000G 9F 9E 0002A
54 0000G CF 9E 00031
50 0000G CF D0 00036
01 0B A0 91 0003B
2C 12 0003F
64 0003F466 8F C9 00041
52 60 B0 00049
22 18 0004C
02 0000FE54 9F D1 0004E
16 18 00055
0000G CF 95 00057
0000G 10 12 0005B
00000000G 9F 0005D
0000G CF 01 FB 00061
01 90 00068
50 D4 0006D 1\$:
04 0006F
01 DD 00070 2\$:
1D DD 00072
6A 02 FB 00074
7E D4 00077
0000G CF 9F 00079
01E4 8F 3C 0007D
00000000G 9F 03 FB 00082

MOVAB INTERRUPT_OCC, R11 ;
MOVAB a#DS\$BGNSUB, R10 ;
MOVAB a#DS\$INLOOP, R9 ;
MOVAB a#DS\$WAITMS, R8 ;
MOVAB a#DS\$ERRHARD, R7 ;
MOVAB a#DS\$SETIPL, R6 ;
MOVAB a#DS\$WAITUS, R5 ;
MOVAB UNIBUS_SPACE, R4 ;
MOVL UBI_LINK, R0 ; 1351
CMPB 11(R0), #1 ;
BNEQ 1\$;
BISL3 #259174, UNIBUS_SPACE, R0 ; 1355
MOVW (R0), TEMP ;
BGEQ 2\$; 1356
CMPL a#^X0000FE54, #2 ; 1359
BGEQ 1\$;
TSTB EXTERNAL_FLAG ;
BNEQ 1\$;
PUSHAB FMT_EXT_PROC ; 1362
CALLS #1, a#DS\$PRINTF ;
MOVB #1, EXTERNAL_FLAG ; 1363
CLRL R0 ; 1365
RET ;
PUSHL #1 ; 1367
PUSHL #29 ;
CALLS #2, DS\$BGNSUB ;
CLRL -(SP) ; 1370
PUSHAB UET_ISR_2 ;
MOVZWL #484, -(SP) ;
CALLS #3, a#DS\$SETVEC ;

; Set the IPL to 1F (hex), set the interrupt expected flag, clear the
; interrupt occurred flag, and initiate an ACLO2 interrupt at B
; R7.Lower the IPL
; from 1F to 17 by steps of one, checking each time that the in
; terrupt has
; not occurred.

53 1F D0 00089
1F DD 0008C
66 01 FB 0008E
64 0003F464 8F C9 00091
60 8000 8F B0 00099
7E 14 7D 0009E
65 02 FB 000A1

MOVL #31, IPL ; 1381
PUSHL #31 ; 1383
CALLS #1, DS\$SETIPL ;
BISL3 #259172, UNIBUS_SPACE, R0 ; 1385
MOVW #-32768, (R0) ; 1386
MOVQ #20, -(SP) ; 1387
CALLS #2, DS\$WAITUS ;

50

50	64	0003F464	8F	C9	000A4	BISL3	#259172, UNIBUS_SPACE, R0	:	
	60	1000	8F	B0	000AC	MOVW	#4096, (R0)	:	1388
	0000G	CF	01	90	000B1	MOVB	#1, INTERRUPT_EXP	:	1389
			6B	94	000B6	CLRB	INTERRUPT_OCC	:	1390
50	64	0003F466	8F	C9	000B8	BISL3	#259174, UNIBUS_SPACE, R0	:	
	60	2000	8F	B0	000C0	MOVW	#8192, (R0)	:	1391
			53	DD	000C5	PUSHL	IPL	:	1393
	66		01	FB	000C7	CALLS	#1, DS\$SETIPL	:	
	7E		14	7D	000CA	MOVQ	#20, -(SP)	:	1394
	65		02	FB	000CD	CALLS	#2, DS\$WAITUS	:	
	1D		6B	E9	000D0	BLBC	INTERRUPT_OCC, 5\$	:	1395
			7E	7C	000D3	CLRQ	-(SP)	:	1400
			53	DD	000D5	PUSHL	IPL	:	
			17	DD	000D7	PUSHL	#23	:	
		0000G	02	DD	000D9	PUSHL	#2	:	
		0000G	CF	9F	000DB	PUSHAB	FMT_UNX_ACINT	:	
		0000G	CF	9F	000DF	PUSHAB	PRINT_GENERAL	:	
		0000G	CF	9F	000E3	PUSHAB	UBIMOD_CSR	:	
	7E	0000G	CF	9A	000E7	MOVZBL	LUN, -(SP)	:	
			01	DD	000EC	PUSHL	#1	:	
			36	11	000EE	BRB	6\$	:	

; Check that the ACLO1 bit is set in CR1.

50	64	0003F464	8F	C9	000F0	5\$: BISL3	#259172, UNIBUS_SPACE, R0	:	1407
	52		60	B0	000F8	MOVW	(R0), TEMP	:	
2A	52		0D	E0	000FB	BBS	#13, TEMP, 7\$	:	1408
			7E	7C	000FF	CLRQ	-(SP)	:	1413
	50		52	3C	00101	MOVZWL	TEMP, R0	:	
7E	50	FFFFDFFF	8F	CB	00104	BICL3	#-8193, R0, -(SP)	:	
	7E	2000	8F	3C	0010C	MOVZWL	#8192, -(SP)	:	
			02	DD	00111	PUSHL	#2	:	
		0000G	CF	9F	00113	PUSHAB	FMT_UET_ACLO1	:	
		0000G	CF	9F	00117	PUSHAB	PRINT_GENERAL	:	
		0000G	CF	9F	0011B	PUSHAB	UBIMOD_CSR	:	
	7E	0000G	CF	9A	0011F	MOVZBL	LUN, -(SP)	:	
			02	DD	00124	PUSHL	#2	:	
	67		0A	FB	00126	6\$: CALLS	#10, DS\$ERRHARD	:	
50	64	0003F464	8F	C9	00129	7\$: BISL3	#259172, UNIBUS_SPACE, R0	:	1414
	60	8000	8F	B0	00131	MOVW	#-32768, (R0)	:	1415
	7E		14	7D	00136	MOVQ	#20, -(SP)	:	1417
	68		02	FB	00139	CALLS	#2, DS\$WAITMS	:	
	69		00	FB	0013C	CALLS	#0, DS\$INLOOP	:	1419
	03		50	E9	0013F	BLBC	R0, 8\$	:	
			FF4C	31	00142	BRW	4\$	:	
FF40	53	FF	8F	17	9D	8\$: ACBB	#23, #-1, IPL, 3\$	:	1381

; Now lower the IPL to 16 and after a nominal wait check to see if the interrupt has occurred. Also check that the ACLO1 bit set in CR1.

50	64	0003F464	8F	C9	0014C	9\$: BISL3	#259172, UNIBUS_SPACE, R0	:	1431
	60	8000	8F	B0	00154	MOVW	#-32768, (R0)	:	1432
			1F	DD	00159	PUSHL	#31	:	1433

	66		01	FB	0015B	CALLS	#1, DS\$SETIPL	:	
50	64	0003F466	8F	C9	0015E	BISL3	#259174, UNIBUS_SPACE, R0	:	
			60	B4	00166	CLRW	(R0)	:	1434
	7E		14	7D	00168	MOVQ	#20, -(SP)	:	1435
	68		02	FB	0016B	CALLS	#2, DS\$WAITMS	:	
0000G	CF		01	90	0016E	MOVB	#1, INTERRUPT_EXP	:	1436
			6B	94	00173	CLRQ	INTERRUPT_OCC	:	1437
			16	DD	00175	PUSHL	#22	:	1438
	66		01	FB	00177	CALLS	#1, DS\$SETIPL	:	
50	64	0003F464	8F	C9	0017A	BISL3	#259172, UNIBUS_SPACE, R0	:	
	60	1000	8F	B0	00182	MOVW	#4096, (R0)	:	1439
50	64	0003F466	8F	C9	00187	BISL3	#259174, UNIBUS_SPACE, R0	:	
	60	2000	8F	B0	0018F	MOVW	#8192, (R0)	:	1440
	7E		14	7D	00194	MOVQ	#20, -(SP)	:	1441
	65		02	FB	00197	CALLS	#2, DS\$WAITUS	:	
	1D		6B	E8	0019A	BLBS	INTERRUPT_OCC, 10\$	:	1442
			7E	7C	0019D	CLRQ	-(SP)	:	1448
			16	DD	0019F	PUSHL	#22	:	
			07	DD	001A1	PUSHL	#7	:	
			02	DD	001A3	PUSHL	#2	:	
		0000G	CF	9F	001A5	PUSHAB	FMT_NO_ACINT	:	
		0000G	CF	9F	001A9	PUSHAB	PRINT_GENERAL	:	
		0000G	CF	9F	001AD	PUSHAB	UBIMOD_CSR	:	
	7E	0000G	CF	9A	001B1	MOVZBL	LUN, -(SP)	:	
			03	DD	001B6	PUSHL	#3	:	
			36	11	001B8	BRB	11\$	:	
; Check that ACLO1 in CR1 <13:00> is set.									
50	64	0003F464	8F	C9	001BA	10\$: BISL3	#259172, UNIBUS_SPACE, R0	:	1456
	52		60	B0	001C2	MOVW	(R0), TEMP	:	
2A	52		0D	E0	001C5	BBS	#13, TEMP, 12\$	:	1457
			7E	7C	001C9	CLRQ	-(SP)	:	1462
	50		52	3C	001CB	MOVZWL	TEMP, R0	:	
7E	50	FFFFDFFF	8F	CB	001CE	BICL3	#-8193, R0, -(SP)	:	
	7E	2000	8F	3C	001D6	MOVZWL	#8192, -(SP)	:	
			02	DD	001DB	PUSHL	#2	:	
		0000G	CF	9F	001DD	PUSHAB	FMT_UET_ACLO1	:	
		0000G	CF	9F	001E1	PUSHAB	PRINT_GENERAL	:	
		0000G	CF	9F	001E5	PUSHAB	UBIMOD_CSR	:	
	7E	0000G	CF	9A	001E9	MOVZBL	LUN, -(SP)	:	
			04	DD	001EE	PUSHL	#4	:	
	67		0A	FB	001F0	11\$: CALLS	#10, DS\$ERRHARD	:	
50	64	0003F466	8F	C9	001F3	12\$: BISL3	#259174, UNIBUS_SPACE, R0	:	1463
	60	8000	8F	B0	001FB	MOVW	#-32768, (R0)	:	1464
	7E		14	7D	00200	MOVQ	#20, -(SP)	:	1465
	65		02	FB	00203	CALLS	#2, DS\$WAITUS	:	
	69		00	FB	00206	CALLS	#0, DS\$INLOOP	:	1467
	03		50	E9	00209	BLBC	R0, 13\$	:	
		FF	3D	31	0020C	BRW	9\$	:	
50	64	0003F464	8F	C9	0020F	13\$: BISL3	#259172, UNIBUS_SPACE, R0	:	
			60	B4	00217	CLRW	(R0)	:	1468
50	64	0003F466	8F	C9	00219	BISL3	#259174, UNIBUS_SPACE, R0	:	
			60	B4	00221	CLRW	(R0)	:	1469
	7E		14	7D	00223	MOVQ	#20,		

```

50      68      0003F464      02 FB 00226      CALLS #2, DS$WAITMS      ;
      64      8000      8F C9 00229      BISL3 #259172, UNIBUS_SPACE, R0      ;
      60      8000      8F B0 00231      MOVW #-32768, (R0)      ; 1471
      66      01 FB 00236      CLRL -(SP)      ; 1472
      01 DD 0023B      CALLS #1, DS$SETIPL      ;
      1D DD 0023D      PUSHL #1      ; 1473
      00000000G 9F 02 FB 0023F      CALLS #2, a#DS$ENDSUB      ;
      02 DD 00246      PUSHL #2      ; 1474
      1D DD 00248      PUSHL #29      ;
      6A      02 FB 0024A      CALLS #2, DS$BGNSUB      ;

```

; Check that INIT clears the interrupt enable bit in CR1 <12:0
0> ACIE

```

50      64 0003F464      8F C9 0024D 14$: BISL3 #259172, UNIBUS_SPACE, R0      ; 1483
      60      8000      8F B0 00255      MOVW #-32768, (R0)      ; 1484
      7E      14 7D 0025A      MOVQ #20, -(SP)      ; 1485
      65      02 FB 0025D      CALLS #2, DS$WAITUS      ;
50      64 0003F464      8F C9 00260      BISL3 #259172, UNIBUS_SPACE, R0      ;
      60      1000      8F B0 00268      MOVW #4096, (R0)      ; 1486
      7E      14 7D 0026D      MOVQ #20, -(SP)      ; 1487
      65      02 FB 00270      CALLS #2, DS$WAITUS      ;
50      64 0003F464      8F C9 00273      BISL3 #259172, UNIBUS_SPACE, R0      ;
      60      8000      8F B0 0027B      MOVW #-32768, (R0)      ; 1488
      7E      14 7D 00280      MOVQ #20, -(SP)      ; 1489
      65      02 FB 00283      CALLS #2, DS$WAITUS      ;
50      64 0003F464      8F C9 00286      BISL3 #259172, UNIBUS_SPACE, R0      ; 1490
      52      60 B0 0028E      MOVW (R0), TEMP      ;
26      52      0C E1 00291      BBC #12, TEMP, 15$      ; 1491
      7E      7C 00295      CLRQ -(SP)      ; 1496
      50      52 3C 00297      MOVZWL TEMP, R0      ;
7E      50 FFFFEEFF      8F CB 0029A      BICL3 #-4097, R0, -(SP)      ;
      7E      02 7D 002A2      MOVQ #2, -(SP)      ;
      0000G      CF 9F 002A5      PUSHAB FMT_INIT_ACLO      ;
      0000G      CF 9F 002A9      PUSHAB PRINT_GENERAL      ;
      0000G      CF 9F 002AD      PUSHAB UBIMOD_CSR      ;
      7E      0000G      CF 9A 002B1      MOVZBL LUN, -(SP)      ;
      05 DD 002B6      PUSHL #5      ;
      67      0A FB 002B8      CALLS #10, DS$ERRHARD      ;
      69      00 FB 002BB 15$: CALLS #0, DS$INLOOP      ; 1498
      8C      50 E8 002BE      BLBS R0, 14$      ;
      02 DD 002C1      PUSHL #2      ; 1499
      1D DD 002C3      PUSHL #29      ;
      00000000G 9F 02 FB 002C5      CALLS #2, a#DS$ENDSUB      ;
      03 DD 002CC      PUSHL #3      ; 1500
      1D DD 002CE      PUSHL #29      ;
      6A      02 FB 002D0      CALLS #2, DS$BGNSUB      ;

```

; Check that the interrupt is inhibited by the interrupt enable bit.

```

50      64 0003F464      8F C9 002D3 16$: BISL3 #259172, UNIBUS_SPACE, R0      ; 1509
      60      8000      8F B0 002DB      MOVW #-32768, (R0)      ; 1510
      7E      14 7D 002E0      MOVQ #20, -(SP)      ; 1511

```

Line	Address	Hex	Op	OpC	OpD	OpE	OpF	OpG	OpH	OpI	OpJ	OpK	OpL	OpM	OpN	OpO	OpP	OpQ	OpR	OpS	OpT	OpU	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH
------	---------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

ZZ-ECCBA-1.4
UBI_TESTS_27_32
01-04

TEST_029: UET CSR1/CSR2 ACLOW Test
TEST_029: UET CSR1/CSR2 ACLOW Test

N 6

6-Sep-1985

Fiche 3

Frame N6

Sequence 490

6-Sep-1985

17:25:18

VAX-11 Bliss-32 V4.1-746

Page 59

6-Sep-1985 17:15:07

[LEMIEUX.ECCBA.RELEASE]ECCBA8.B32;19 (5)

		0000G	CF	9F	003A3	PUSHAB	PRINT_GENERAL	:
		0000G	CF	9F	003A7	PUSHAB	UBIMOD_CSR	:
	7E	0000G	CF	9A	003AB	MOVZBL	LUN, -(SP)	:
			07	DD	003B0	PUSHL	#7	:
	67		0A	FB	003B2	CALLS	#10, DS\$ERRHARD	:
	69		00	FB	003B5	CALLS	#0, DS\$INLOOP	: 1553
	8C		50	E8	003B8	BLBS	R0, 18\$	:
			04	DD	003BB	PUSHL	#4	: 1554
			1D	DD	003BD	PUSHL	#29	:
00000000G	9F		02	FB	003BF	CALLS	#2, a#DS\$ENDSUB	:
	7E	01E4	8F	3C	003C6	MOVZWL	#484, -(SP)	: 1560
00000000G	9F		01	FB	003CB	CALLS	#1, a#DS\$CLRVEC	:
00000000G	9F		00	FB	003D2	CALLS	#0, a#DS\$BREAK	:
	50		01	D0	003D9	MOVL	#1, R0	: 1564
			04	003DC	RET			:

; Routine Size: 989 bytes, Routine Base: TEST_029 + 0000

; 1565 1

```

; 1566 2 $DS_BGNTST (SECTION = (DEFAULT,ALL), TEXT = 'Map Invalid Test');
; 1567 2
; 1568 2
; 1569 2 !**
; 1570 2 ! TEST DESCRIPTION:
; 1571 2 !
; 1572 2 !     This test checks that UBI refuses to issue SSYN if the UET attempts
; 1573 2 !     to access a map entry which has the valid bit turned off.
; 1574 2 !
; 1575 2 ! ASSUMPTIONS:
; 1576 2 !
; 1577 2 !     Test 1 - Test 15
; 1578 2 !--
; 1579 2 LOCAL
; 1580 2     BUF_PFN,
; 1581 2     MAP_NUMBER,
; 1582 2     N,
; 1583 2     UBUS_ADDR: WORD,
; 1584 2     TEMP,
; 1585 2     TEMP3;
; 1586 2
; 1587 2 CODECOMMENT
; 1588 2 ''
; 1589 2 'Set up the UET for a transfer to a map which has the valid bit cleared.',
; 1590 2 'First set up the UET to point to map entry 0, and then clear map entry 0.',
; 1591 2 ''
; 1592 3 BEGIN
; 1593 3
; 1594 3     UET_ADDR_REG = 0;
; 1595 3
; 1596 3     TEMP3 = 0;
; 1597 3
; 1598 3     UBI_MAP(0) = 0;
; 1599 3
; 1600 3 END;
; 1601 2
; 1602 2 CODECOMMENT
; 1603 2 ''
; 1604 2 'Execute a DATI, and after a nominal wait check that the UET shows NXM',
; 1605 2 'status.',
; 1606 2 ''
; 1607 3 BEGIN
; 1608 3
; 1609 3     UET_CTRL_REG = .TEMP3 OR DATI;
; 1610 3
; 1611 3     $DS_WAITMS(TIME=10);
; 1612 3
; 1613 3     TEMP = .UET_CTRL_REG;
; 1614 3     IF NOT .TEMP<6,1>
; 1615 3
; 1616 3 THEN
; 1617 4 BEGIN
; 1618 4     $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_DDP,
; 1619 4     PRLINK = PRINT_GENERAL,
; 1620 4     P1 = FMT_MAP_INV,P2 = 0);

```

ZZ-ECCBA-1.4
UBI_TESTS_27_32
01-04

TEST_030: Map Invalid Test

TEST_030: Map Invalid Test

Test 30, Subtest 0, Error 1

END;

END;

UET_CTRL_REG = %X'8000';

\$DS_ENDTEST;

C 7

6-Sep-1985

Fiche 3 Frame C7

Sequence 492

6-Sep-1985 17:25:18

VAX-11 Bliss-32 V4.1-746

Page 61

6-Sep-1985 17:15:07

[LEMIEUX.ECCBA.RELEASE]ECCBA8.B32;19

(6)

; xPRINT:
; 1621 3
; 1622 3
; 1623 2
; 1624 2
; 1625 2
; 1626 2
; 1627 1

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00048 \$:
00000000 00000003 00058

.ADDRESS P.AAJ, P.AAK, P.AAL, TEST_030
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

65 54 20 64 69 6C 61 76 6E 49 20 70 61 4D 10 001E 00060 P.AAJ: .WORD 30
74 73 00062 P.AAK: .ASCII <16>\Map Invalid Test\
00071
00073
00000000 00074 P.AAL: .BLKB 1
.LONG 0

.PSECT TEST_030,NOWRT,2

52 0000G CF 0004 00000
9E 00002

.ENTRY TEST_030, Save R2
MOVAB UNIBUS_SPACE, R2

; 1566
;

; Set up the UET for a transfer to a map which has the valid bit cleared.
; First set up the UET to point to map entry 0, and then clear map entry 0.

50 62 0003F460 8F C9 00007
60 B4 0000F
51 D4 00011
50 0000G CF D0 00013
0800 C0 D4 00018

BISL3 #259168, UNIBUS_SPACE, R0
CLRW (R0)
CLRL TEMP3
MOVL UBI_UNDER_TEST, R0
CLRL 2048(R0)

; 1592
; 1594
; 1596
; 1598
;

; Execute a DATI, and after a nominal wait check that the UET shows NXM status.

50 62 0003F464 8F C9 0001C
60 51 A9 00024
7E 0A 7D 00028
00000000G 9F 02 FB 0002B
50 62 0003F464 8F C9 00032
50 60 3C 0003A
20 50 06 E0 0003D
7E 7C 00041
7E 7C 00043

BISL3 #259172, UNIBUS_SPACE, R0
BISW3 #1, TEMP3, (R0)
MOVQ #10, -(SP)
CALLS #2, @DS\$WAITMS
BISL3 #259172, UNIBUS_SPACE, R0
MOVZWL (R0), TEMP
BBS #6, TEMP, 1\$
CLRQ -(SP)
CLRQ -(SP)

; 1607
; 1609
; 1611
;
; 1613
;
; 1614
; 1620
;

ZZ-ECCBA-1.4 TEST_030: Map Invalid Test
UBI TESTS_27_32
01-04 TEST_030: Map Invalid Test

D 7
6-Sep-1985 Fiche 3 Frame D7 Sequence 493
6-Sep-1985 17:25:18 VAX-11 Bliss-32 V4.1-746 Page 62
6-Sep-1985 17:15:07 [LEMIEUX.ECCBA.RELEASE]ECCBA8.B32;19 (6)

			7E	D4	00045	CLRL	-(SP)	:	
		0000G	CF	9F	00047	PUSHAB	FMT_MAP_INV	:	
		0000G	CF	9F	0004B	PUSHAB	PRINT_GENERAL	:	
		0000G	CF	9F	0004F	PUSHAB	UBIMOD_DDP	:	
		0000G	CF	9A	00053	MOVZBL	LUN, -(SP)	:	
			01	DD	00058	PUSHL	#1	:	
	00000000G	9F	0A	FB	0005A	CALLS	#10, @DS\$ERRHARD	:	
50		62	0003F464	8F	C9	00061	BISL3	#259172, UNIBUS_SPACE, R0	: 1623
		60	8000	8F	B0	00069	MOVW	#-32768, (R0)	: 1625
	00000000G	9F		00	FB	0006E	CALLS	#0, @DS\$BREAK	:
		50		01	D0	00075	MOVL	#1, R0	: 1627
				04	00078	RET		:	

; Routine Size: 121 bytes, Routine Base: TEST_030 + 0000

```
; 1628 2 $DS_BGNTEST (SECTION=(DEFAULT,ALL),TEXT='UET PB Bit Test');
; 1629 2
; 1630 2
; 1631 2 !+
; 1632 2 ! TEST DESCRIPTION:
; 1633 2 !
; 1634 2 ! This test checks that if PB is ascerted on the Unibus, the CPU will
; 1635 2 ! machine check if it tries to read the Unibus.
; 1636 2 !
; 1637 2 ! ASSUMPTIONS:
; 1638 2 !
; 1639 2 ! Tests 1 thru 28
; 1640 2 !--
; 1641 2 UET_CTRL_REG = %X'20'; ! Set the PB bit in the UET
; 1642 2 IF PROBE_ADDRESS(UET_DATA_REG,0,0)
; 1643 2 THEN
; 1644 3 BEGIN
; P 1645 3 $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD,
; P 1646 3 PRLINK = PRINT_GENERAL,
; L 1647 3 P1 = FMT_NO_MCHK,P2 = 0);
; %PRINT: Test 31, Subtest 0, Error 1
; 1648 2 END;
; 1649 2
; 1650 2 CODECOMMENT
; 1651 2 ''
; 1652 2 'Check that the PB bit clears with a UET INIT.'
; 1653 2 ''
; 1654 3 BEGIN
; 1655 3 UET_CTRL_REGB = %X'80';
; 1656 3 IF NOT PROBE_ADDRESS(UET_CTRL_REG,0,0)
; 1657 3 THEN
; 1658 4 BEGIN
; P 1659 4 $DS_ERRHARD(UNIT=.LUN,MSGADR=UETMOD,
; P 1660 4 PRLINK = PRINT_GENERAL,
; L 1661 4 P1 = FMT_UET_PB,P2 = 0);
; %PRINT: Test 31, Subtest 0, Error 2
; 1662 3 END;
; 1663 2 END;
; 1664 2
; 1665 1 $DS_ENDTEST;
```

.PSECT DISPATCH,NOWRT,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00060 \$:
00000000 00000003 00070

.ADDRESS P.AAM, P.AAN, P.AAO, TEST_031
.LONG 3, 0

.PSECT \$PLIT\$,NOWRT,NOEXE,2

73 65 54 20 74 69 42 20 42 50 20 54 45 55 0F 74
001F 00078 P.AAM:
0007A P.AAN:
00089
0008A
00000000 0008C P.AAO:

.WORD 31
.ASCII <15>\UET PB Bit Test\
.BLKB 2
.LONG 0

```

                                .PSECT TEST_031,NOWRT,2
                                .ENTRY TEST_031, Save R2,R3
                                MOVAB UNIBUS_SPACE, R3
                                MOVAB @DS$ERRHARD, R2
                                BISL3 #259172, UNIBUS_SPACE, R0
                                MOVW #32, (R0)
                                CLRQ -(SP)
                                BISL3 #259170, UNIBUS_SPACE, -(SP)
                                CALLS #3, PROBE_ADDRESS
                                BLBC R0, 1$
                                CLRQ -(SP)
                                CLRQ -(SP)
                                CLRL -(SP)
                                PUSHAB FMT_NO_MCHK
                                PUSHAB PRINT_GENERAL
                                PUSHAB UBIMOD
                                MOVZBL LUN, -(SP)
                                PUSHL #1
                                CALLS #10, DS$ERRHARD
                                ;
                                ; Check that the PB bit clears with a UET INIT.
                                ;
                                1$: BISL3 #259173, UNIBUS_SPACE, R0
                                MOVW #-128, (R0)
                                CLRQ -(SP)
                                BISL3 #259172, UNIBUS_SPACE, -(SP)
                                CALLS #3, PROBE_ADDRESS
                                BLBS R0, 2$
                                CLRQ -(SP)
                                CLRQ -(SP)
                                CLRL -(SP)
                                PUSHAB FMT_UET_PB
                                PUSHAB PRINT_GENERAL
                                PUSHAB UETMOD
                                MOVZBL LUN, -(SP)
                                PUSHL #2
                                CALLS #10, DS$ERRHARD
                                CALLS #0, @DS$BREAK
                                MOVL #1, R0
                                RET

```

; Routine Size: 140 bytes, Routine Base: TEST_031 + 0000


```
; 1666 2 $DS_BGNTST (SECTION=(DEFAULT,ALL,UBE),TEXT='UBE Block Transfer Test');
; 1667 2
; 1668 2 !**
; 1669 2 ! TEST DESCRIPTION:
; 1670 2 !
; 1671 2 ! This test supports up to four UBEs, each of which is assigned a UBI
; 1672 2 ! data path. Each UBE is set up to perform large block transfers at
; 1673 2 ! high speed (10 microseconds per word). All transfers are started
; 1674 2 ! together, such that a maximum of bus activity takes place. The
; 1675 2 ! program selects the appropriate routine to use based on the number of
; 1676 2 ! UBEs selected (the user determines the number of UBEs to use, along
; 1677 2 ! with their addresses and vectors).
; 1678 2 !
; 1679 2 ! While waiting for the transfers to complete, each unused map is
; 1680 2 ! written with zero's and one's and read and checked.
; 1681 2 !
; 1682 2 ! ASSUMPTIONS:
; 1683 2 !
; 1684 2 ! Test 1-Test 24
; 1685 2 !--
; 1686 2
; 1687 2 LOCAL
; 1688 2 I,
; 1689 2 MAP_NUM,
; 1690 2 MAP_DATA,
; 1691 2 TEMP0: WORD,
; 1692 2 TEMP1: WORD,
; 1693 2 TEMP2: WORD,
; 1694 2 TEMP3: WORD,
; 1695 2 TEMP4: WORD,
; 1696 2 TEMP5,
; 1697 2 TEMP6,
; 1698 2 TEMP7,
; 1699 2 WORD_PATTERN_0: VECTOR[4,WORD],
; 1700 2 WORD_PATTERN_1: VECTOR[4,WORD],
; 1701 2 WORD_PATTERN_2: VECTOR[4,WORD],
; 1702 2 WORD_PATTERN_3: VECTOR[4,WORD],
; 1703 2 NO_MAPS_INUSE, ! A10
; 1704 2 CURRENT_MAP; ! A10
; 1705 2
; 1706 2 LITERAL
; 1707 2 DATO_BR4 = %X'663', ! Redefine DATO for UBE (not UET) ! A8
; 1708 2 DATO_BR5 = %X'665',
; 1709 2 DATO_BR6 = %X'669',
; 1710 2 DATO_BR7 = %X'671';
; 1711 2
; 1712 2 MACRO
; M 1713 2 UBE_DB(N)=
; 1714 2 (.UBE_BASE_ADDR[N])<0,16>%, ! UBE data buffer
; M 1715 2 UBE_CC(N)=
; 1716 2 (.UBE_BASE_ADDR[N] + 2)<0,16>%, ! UBE cycle count register
; M 1717 2 UBE_BA(N)=
; 1718 2 (.UBE_BASE_ADDR[N] + 4)<0,16>%, ! UBE bus address register
; M 1719 2 UBE_CR1(N)=
; 1720 2 (.UBE_BASE_ADDR[N] + 6)<0,16>%, ! UBE control register no. 1
```

```

; M 1721 2      UBE_CE(N)=
; 1722 2      (.UBE_BASE_ADDR[N] + 8)<0,16>x, ! UBE clear error register
; M 1723 2      UBE_SG(N)=
; 1724 2      (.UBE_BASE_ADDR[N] + 12)<0,16>x, ! UBE simultaneous go register
; M 1725 2      UBE_CR2(N)=
; 1726 2      (.UBE_BASE_ADDR[N] + 14)<0,16>x, ! UBE control register no. 2
; M 1727 2      UBE_DUMP(N)=
; 1728 2      (
; M 1729 2      TEMP0 = .UBE_DB(N);
; M 1730 2      TEMP1 = .UBE_CC(N);
; M 1731 2      TEMP2 = .UBE_BA(N);
; M 1732 2      TEMP3 = .UBE_CR1(N);
; M 1733 2      TEMP4 = .UBE_CR2(N);
; M 1734 2      $DS_PRINTX(FMT_UBE_DUMP,.UBE_DRIVE[N],.TEMP0,.TEMP1,
; M 1735 2      .TEMP2,.TEMP3,.TEMP4);
; 1736 2      )x,
; M 1737 2      UBI_DUMP=
; 1738 2      (
; M 1739 2      TEMP4 = .UBI_CSR(0) AND CSR_MASK;
; M 1740 2      TEMP5 = .UBI_CSR(1) AND CSR_MASK;
; M 1741 2      TEMP6 = .UBI_CSR(2) AND CSR_MASK;
; M 1742 2      TEMP7 = .UBI_CSR(3) AND CSR_MASK;
; M 1743 2      $DS_PRINTX(FMT_UBI_DUMP,.TEMP4,.TEMP5,.TEMP6,.TEMP7);
; 1744 2      )x;
; 1745 2
; 1746 2      !$DS_SETVEC(VECTOR=xx'348',SRVADR=UBE0_ISR);
; 1747 2
; 1748 3      IF ((.NUM_UBE EQL 0) AND (.DSA$GL_PASSNO LSS 2))
; 1749 3      THEN $DS_PRINTX(FMT_NO_UBE)
; 1750 2      ELSE
; 1751 3      BEGIN
; 1752 3      INCR I FROM 0 TO .NUM_UBE - 1 DO
; 1753 4      BEGIN
; 1754 4      IF NOT PROBE_ADDRESS(UBE_DB(.I),0,0)
; 1755 4      THEN
; 1756 5      BEGIN
; P 1757 5      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD,
; P 1758 5      PRLINK = PRINT_GENERAL,
; P 1759 5      P1 = FMT_UB_TO,P2 = 1,
; L 1760 5      P3 = UBE_DB(.I));
; xPRINT:      Test 32, Subtest 0, Error 1
; 1761 4      END;
; 1762 3      END;
; 1763 2      END;
; 1764 2
; 1765 2      CODECOMMENT
; 1766 2      'Set up the data patterns to be used in the test.'
; 1767 2      '
; 1768 2      '
; 1769 3      BEGIN
; 1770 3
; 1771 3      WORD_PATTERN_0[0] = xx'5555';
; 1772 3      WORD_PATTERN_0[1] = xx'AAAA';
; 1773 3      WORD_PATTERN_0[2] = xx'3333';
; 1774 3      WORD_PATTERN_0[3] = xx'0F0F';

```

! D8

! A8

! A8

! A8

! A8

! A8 M11

! A8

! A8

! A8

! A8

! A8

! A8

! A8

! A8

```

: 1775 3
: 1776 3      WORD_PATTERN_1[0] = XX'1234';
: 1777 3      WORD_PATTERN_1[1] = XX'5678';
: 1778 3      WORD_PATTERN_1[2] = XX'ABCD';
: 1779 3      WORD_PATTERN_1[3] = XX'4444';
: 1780 3
: 1781 3      WORD_PATTERN_2[0] = XX'AAAA';
: 1782 3      WORD_PATTERN_2[1] = XX'5555';
: 1783 3      WORD_PATTERN_2[2] = XX'6666';
: 1784 3      WORD_PATTERN_2[3] = XX'F0F0';
: 1785 3
: 1786 3      WORD_PATTERN_3[0] = XX'2398';
: 1787 3      WORD_PATTERN_3[1] = XX'1764';
: 1788 3      WORD_PATTERN_3[2] = XX'9987';
: 1789 3      WORD_PATTERN_3[3] = XX'EEEE';
: 1790 3
: 1791 2      END;
: 1792 2
: 1793 2      CODECOMMENT
: 1794 2      'Determine how many UBEs have been selected for test by the user.'
: 1795 2      '
: 1796 2      '
: 1797 3      BEGIN
: 1798 3      CASE .NUM_UBE FROM 1 TO 4 OF
: 1799 3      SET
: 1800 3
: 1801 3      [1]:
: 1802 4      BEGIN
: 1803 4
: 1804 4      INCR I FROM 0 TO 1 DO      ! Normal and byte offset A10
: 1805 5      BEGIN
: 1806 5      UBE_MODE[0] = .I;          ! A10
: 1807 5
: 1808 5      UBE_CE(0) = 0;            ! A8
: 1809 5
: 1810 5      CODECOMMENT
: 1811 5      '
: 1812 5      'This is the action taken for one UBE. First, the map buffers',
: 1813 5      'routine is called. This routine allocates buffer space for the',
: 1814 5      'transfers.',
: 1815 5      '
: 1816 6      BEGIN
: 1817 6      NO_MAPS_INUSE = MAP_BUFFERS(1,.UBE_MODE[0]);      ! M10
: 1818 6      IF .NO_MAPS_INUSE EQL 0
: 1819 6      THEN $DS_ABORT(ARG = TEST);
: 1820 5      END;
: 1821 5
: 1822 5      INCR N FROM 0 TO 3 DO      ! Data pattern selector
: 1823 6      BEGIN
: 1824 6
: 1825 6      CODECOMMENT
: 1826 6      '
: 1827 6      'Next, set up the buffer. The parameters are buffer number',
: 1828 6      'and the pattern to use.',
: 1829 6      '

```



```

; 1830 7      BEGIN
; 1831 7      BUFFER_SETUP(0,.WORD_PATTERN_0[.N],.UBE_MODE[0]);      ! M10
; 1832 7      UBE_INIT_DATA[0] = .WORD_PATTERN_0[.N];      ! A10
; 1833 6      END;
; 1834 6
; 1835 6      CODECOMMENT
; 1836 6      'Set up the UBE and UBI for the transfer. The parameters',
; 1837 6      'are UBE number, starting map number, offset mode, data path',
; 1838 6      'number, rotate data path mode.',
; 1839 6      '':
; 1840 6      BEGIN
; 1841 7      IF .UBE_MODE[0] EQL 0
; 1842 7      THEN
; 1843 7      BEGIN
; 1844 8      XFER_SETUP(0,.UBE_MAP_NO[0],0,1);
; 1845 8      UBE_DP_NO[0] = -1;      ! A10
; 1846 8      END
; 1847 8      ELSE
; 1848 7      BEGIN
; 1849 8      XFER_SETUP(0,.UBE_MAP_NO[0],0,0);
; 1850 8      UBE_DP_NO[0] = 0;
; 1851 8      END;
; 1852 7      END;
; 1853 6
; 1854 6      CODECOMMENT
; 1855 6      'Set up the UBE data buffer and the expected data for the',
; 1856 6      'transfer analysis routine. Set the IPL to 16 to allow the',
; 1857 6      'UBE to interrupt on error. Then start the transfer.',
; 1858 6      '':
; 1859 6      BEGIN
; 1860 7      $DS_BREAK;
; 1861 7      UBE_STAT_ERR[0] = 0;      ! A8
; 1862 7      UBE_XFER_ERR[0] = 0;      ! A8
; 1863 7      UBE_DB[0] = NOT .WORD_PATTERN_0[.N];
; 1864 7      UBE_EXP_DATA[0] = NOT .WORD_PATTERN_0[.N];
; 1865 7      $DS_SETIPL(LEVEL=XX'16');      ! D8
; 1866 7      UBE_CR1[0] = DATO_BR4;
; 1867 7      !
; 1868 7      END;
; 1869 6
; 1870 6      CODECOMMENT
; 1871 6      'Wait for the transfer to complete.',
; 1872 6      'then check to see if a status error was indicated. If it',
; 1873 6      'was, then dump the UBI and UBE registers. If there a',
; 1874 6      'status error was not indicated, then check to see if there',
; 1875 6      'was a data error. If so, report it.',
; 1876 6      '':
; 1877 6      BEGIN
; 1878 7      $DS_WAITMS(TIME=10);      ! D8
; 1879 7      UBE_INT_OCC = 0;      ! A8
; 1880 7      CURRENT_MAP = .NO_MAPS_INUSE - 1;      ! A10
; 1881 7      WHILE 1 DO      ! A8
; 1882 7      BEGIN
; 1883 7
; 1884 8

```

```

: 1885 8      IF .UBE_INT_OCC[0]      ! A8
: 1886 8      THEN EXITLOOP;          ! A8
: 1887 8      IF .CURRENT_MAP EQL 511  ! A10
: 1888 8      THEN CURRENT_MAP = .NO_MAPS_INUSE ! A10
: 1889 8      ELSE CURRENT_MAP = .CURRENT_MAP + 1; ! A10
: 1890 8
: 1891 8      !
: 1892 8      ! Write and Read the Map Register
: 1893 8      !
: 1894 8      UBI_MAP(.CURRENT_MAP) = 0; ! A10
: 1895 8      IF (MAP_DATA = (.UBI_MAP(.CURRENT_MAP) AND MAP_MASK)) NEQ 0 ! A10
: 1896 8      THEN
: 1897 9          BEGIN
: 1898 9              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,PRLINK=PRINT_MAP_ERR,! A10 M13
: 1899 9              P1=.CURRENT_MAP,P2=UBI_MAP(.CURRENT_MAP),
: 1900 9              P3=FMT_MAP_FAIL,P4=0,P5=.MAP_DATA);
: 1901 9      Test 32, Subtest 0, Error 2
: 1902 9      !
: 1903 9      $DS_PRINTX(FMT_MAP_FAIL1,.CURRENT_MAP,0,.MAP_DATA); ! A10 D13
: 1904 9      END
: 1905 9      ELSE
: 1906 9      BEGIN
: 1907 9          UBI_MAP(.CURRENT_MAP) = -1; ! A10
: 1908 9          IF (MAP_DATA = (.UBI_MAP(.CURRENT_MAP) AND MAP_MASK)) NEQ MAP_MASK ! A10
: 1909 9          THEN
: 1910 9              BEGIN
: 1911 9                  $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,PRLINK=PRINT_MAP_ERR,! A10 M13
: 1912 9                  P1=.CURRENT_MAP,P2=UBI_MAP(.CURRENT_MAP),
: 1913 9                  P3=FMT_MAP_FAIL,P4=MAP_MASK,P5=.MAP_DATA);
: 1914 9      Test 32, Subtest 0, Error 3
: 1915 9      !
: 1916 9      $DS_PRINTX(FMT_MAP_FAIL1,.CURRENT_MAP,MAP_MASK,.MAP_DATA); ! A10 D13
: 1917 9      END;
: 1918 9      END;
: 1919 9      IF .UBE_STAT_ERR[0]
: 1920 9      THEN
: 1921 9      BEGIN
: 1922 9          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP); ! M7
: 1923 9      Test 32, Subtest 0, Error 4
: 1924 9      UBI_DUMP;
: 1925 9      UBE_DUMP(0);
: 1926 9      END
: 1927 9      ELSE
: 1928 9      BEGIN
: 1929 9          IF .UBE_XFER_ERR[0]
: 1930 9          THEN
: 1931 9              $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP, ! M7
: 1932 9              PRLINK=PRINT_DCOMP_ERR,P1=0);
: 1933 9      Test 32, Subtest 0, Error 5
: 1934 9      END;
: 1935 9      END;
: 1936 9      END;
: 1937 9      $DS_RELBUF(PAGCNT=.UBE_PAGE_CNT,RETADR=UBE_BUF_ADDR[0]); ! A8
: 1938 9      END;
: 1939 9      ! INCR N
: 1940 9      ! INCR I
: 1941 9      ! CASE
: 1942 9      [2]:

```

```

: 1936 4      BEGIN
: 1937 4
: 1938 4      INCR I FROM 0 TO 3 DO      ! Select Data Paths      ! A10
: 1939 5      BEGIN
: 1940 5      UBE_DP_NO[0] = .I;          ! A10
: 1941 5      IF .UBE_DP_NO[0] EQL 3      ! A10
: 1942 5      THEN UBE_DP_NO[1] = 0      ! A10
: 1943 5      ELSE UBE_DP_NO[1] = .UBE_DP_NO[0] + 1;      ! A10
: 1944 5
: 1945 5      UBE_CE(0) = 0;              ! A8
: 1946 5      UBE_CE(1) = 0;              ! A8
: 1947 5
: 1948 5      UBE_MODE[0] = 0;            ! A10
: 1949 5      UBE_MODE[1] = 1;            ! A10
: 1950 5
: 1951 5      CODECOMMENT
: 1952 5      ''
: 1953 5      'This is the action taken for two UBEs. First, the map buffers',
: 1954 5      'routine is called. This routine allocates buffer space for the',
: 1955 5      'transfers.',
: 1956 5      ''
: 1957 6      BEGIN
: 1958 6      NO_MAPS_INUSE = MAP_BUFFERS(2,1);      ! M10
: 1959 6      IF .NO_MAPS_INUSE EQL 0
: 1960 6      THEN $DS_ABORT(ARG = TEST);
: 1961 5      END;
: 1962 5
: 1963 5      INCR N FROM 0 TO 3 DO      ! Data pattern select
: 1964 6      BEGIN
: 1965 6
: 1966 6      CODECOMMENT
: 1967 6      ''
: 1968 6      'Next, set up the buffer. The parameters are buffer number',
: 1969 6      'and the pattern to use.',
: 1970 6      ''
: 1971 7      BEGIN
: 1972 7      BUFFER_SETUP(0,.WORD_PATTERN_0[.N],0);
: 1973 7      BUFFER_SETUP(1,.WORD_PATTERN_1[.N],1);
: 1974 7      UBE_INIT_DATA[0] = .WORD_PATTERN_0[.N];      ! A10
: 1975 7      UBE_INIT_DATA[1] = .WORD_PATTERN_1[.N];      ! A10
: 1976 6      END;
: 1977 6
: 1978 6      CODECOMMENT
: 1979 6      ''
: 1980 6      'Set up the UBEs and UBI for the transfer. The parameters',
: 1981 6      'are UBE number, starting map number, offset mode, data path',
: 1982 6      'number, rotate data path mode, and data pattern if DATIP.',
: 1983 6      ''
: 1984 7      BEGIN
: 1985 7      XFER_SETUP(0,.UBE_MAP_NO[0],.UBE_DP_NO[0],0);      ! M10
: 1986 7      XFER_SETUP(1,.UBE_MAP_NO[1],.UBE_DP_NO[1],0);      ! M10
: 1987 6      END;
: 1988 6
: 1989 6      CODECOMMENT
: 1990 6      ''

```



```

: 1991 6      'Set up the UBE data buffer and the expected data for the',
: 1992 6      'transfer analysis routine. Set the IPL to 16 to allow the',
: 1993 6      'UBE to interrupt on error. Then start the transfer.',
: 1994 6      '';
: 1995 7      BEGIN
: 1996 7      LOCAL X;
: 1997 7      $DS_BREAK;
: 1998 7      INCR X FROM 0 TO 1 DO
: 1999 8      BEGIN
: 2000 8      UBE_STAT_ERR[.X] = 0;
: 2001 8      UBE_XFER_ERR[.X] = 0;
: 2002 7      END;
: 2003 7      UBE_DB(0) = NOT .WORD_PATTERN_0[.N];
: 2004 7      UBE_DB(1) = NOT .WORD_PATTERN_1[.N];
: 2005 7      UBE_EXP_DATA[0] = NOT .WORD_PATTERN_0[.N];
: 2006 7      UBE_EXP_DATA[1] = NOT .WORD_PATTERN_1[.N];
: 2007 7      $DS_SETIPL(LEVEL=X'16');
: 2008 7      UBE_CR1(0) = DATO_BR4;
: 2009 7      UBE_CR1(1) = DATO_BR5;
: 2010 6      END;
: 2011 6
: 2012 6      CODECOMMENT
: 2013 6      '';
: 2014 6      'Wait for enough time to allow the transfer to complete.',
: 2015 6      'then check to see if a status error was indicated. If it',
: 2016 6      'was, then dump the UBI and UBE registers. If there a',
: 2017 6      'status error was not indicated, then check to see if there',
: 2018 6      'was a data error. If so, report it.',
: 2019 6      '';
: 2020 7      BEGIN
: 2021 7      $DS_WAITMS(TIME=20);
: 2022 7      UBE_INT_OCC = 0;
: 2023 7      CURRENT_MAP = .NO_MAPS_INUSE - 1;
: 2024 7      WHILE 1 DO
: 2025 8      BEGIN
: 2026 9      IF (.UBE_INT_OCC[0] AND .UBE_INT_OCC[1])
: 2027 8      THEN EXITLOOP;
: 2028 8      IF .CURRENT_MAP EQL 511
: 2029 8      THEN CURRENT_MAP = .NO_MAPS_INUSE
: 2030 8      ELSE CURRENT_MAP = .CURRENT_MAP + 1;
: 2031 8
: 2032 8      !
: 2033 8      ! Write and Read the Map Register
: 2034 8      !
: 2035 8      UBI_MAP(.CURRENT_MAP) = 0;
: 2036 8      IF (MAP_DATA = (.UBI_MAP(.CURRENT_MAP) AND MAP_MASK)) NEQ 0
: 2037 8      THEN
: 2038 9      BEGIN
: 2039 9      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,PRLINK=PRINT_MAP_ERR,! A10 M13
: 2040 9      P1=.CURRENT_MAP,P2=UBI_MAP(.CURRENT_MAP),
: 2041 9      P3=FMT_MAP_FAIL,P4=0,P5=.MAP_DATA);
: 2042 9      $DS_PRINTX(FMT_MAP_FAIL1,.CURRENT_MAP,0,.MAP_DATA);! A10 D13
: 2043 9      END
: 2044 8      ELSE

```

Test 32, Subtest 0, Error 6

```

: 2045 9      BEGIN
: 2046 9      UBI_MAP(.CURRENT_MAP) = -1;          ! A10
: 2047 9      IF (MAP_DATA = (.UBI_MAP(.CURRENT_MAP) AND MAP_MASK)) NEQ MAP_MASK ! A10
: 2048 9      THEN
: 2049 10      BEGIN
: P 2050 10      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,PRLINK=PRINT_MAP_ERR,! A10 M13
: P 2051 10      P1=.CURRENT_MAP,P2=UBI_MAP(.CURRENT_MAP),
: L 2052 10      P3=FMT_MAP_FAIL,P4=MAP_MASK,P5=.MAP_DATA);
: xPRINT:      Test 32, Subtest 0, Error 7
: 2053 10      !
: 2054 9      $DS_PRINTX(FMT_MAP_FAIL1,.CURRENT_MAP,MAP_MASK,.MAP_DATA);! A10 D13
: 2055 8      END;
: 2056 7      END;
: 2057 7      IF .UBE_STAT_ERR[0]
: 2058 7      THEN
: 2059 8      BEGIN
: L 2060 8      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP);          ! M7
: xPRINT:      Test 32, Subtest 0, Error 8
: 2061 8      UBI_DUMP;
: 2062 8      UBE_DUMP(0);
: 2063 8      END
: 2064 7      ELSE
: 2065 8      BEGIN
: 2066 8      IF .UBE_XFER_ERR[0]
: 2067 8      THEN
: P 2068 8      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,          ! M7
: L 2069 8      PRLINK=PRINT_DCMP_ERR,P1=0);
: xPRINT:      Test 32, Subtest 0, Error 9
: 2070 7      END;
: 2071 7      IF .UBE_STAT_ERR[1]
: 2072 7      THEN
: 2073 8      BEGIN
: L 2074 8      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP);          ! M7
: xPRINT:      Test 32, Subtest 0, Error 10
: 2075 8      UBI_DUMP;
: 2076 8      UBE_DUMP(1);
: 2077 8      END
: 2078 7      ELSE
: 2079 8      BEGIN
: 2080 8      IF .UBE_XFER_ERR[1]
: 2081 8      THEN
: P 2082 8      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,          ! M7
: L 2083 8      PRLINK=PRINT_DCMP_ERR,P1=1);
: xPRINT:      Test 32, Subtest 0, Error 11
: 2084 7      END;
: 2085 6      END;
: 2086 5      END;          ! INCR N
: 2087 5      $DS_RELBUF(PAGCNT=.UBE_PAGE_CNT);          ! A8
: 2088 4      END;          ! INCR I
: 2089 3      END;          ! CASE
: 2090 3      [3]:
: 2091 4      BEGIN
: 2092 4
: 2093 4      INCR I FROM 0 TO 3 DO          ! Data path select          ! A10
: 2094 5      BEGIN

```

```

: 2095 5      UBE_DP_NO[0] = .1;          ! A10
: 2096 5      IF .UBE_DP_NO[0] EQL 3      ! A10
: 2097 5      THEN UBE_DP_NO[1] = 0      ! A10
: 2098 5      ELSE UBE_DP_NO[1] = .UBE_DP_NO[0] + 1; ! A10
: 2099 5      IF .UBE_DP_NO[1] EQL 3      ! A10
: 2100 5      THEN UBE_DP_NO[2] = 0      ! A10
: 2101 5      ELSE UBE_DP_NO[2] = .UBE_DP_NO[1] + 1; ! A10
: 2102 5
: 2103 5      UBE_MODE[0] = 1;            ! A10
: 2104 5      UBE_MODE[1] = 0;            ! A10
: 2105 5      UBE_MODE[2] = 1;            ! A10
: 2106 5
: 2107 6      BEGIN
: 2108 6      LOCAL X;
: 2109 6      INCR X FROM 0 TO 2 DO      ! A8
: 2110 6          UBE_CE(.X) = 0;        ! A8
: 2111 5      END;
: 2112 5
: 2113 5      CODECOMMENT
: 2114 5      'This is the action taken for three UBEs. First, the map',
: 2115 5      'buffers routine is called. This routine allocates buffer space',
: 2116 5      'for the transfers.',
: 2117 5      '':
: 2118 5      BEGIN
: 2119 6      NO_MAPS_INUSE = MAP_BUFFERS(3,1); ! M10
: 2120 6      IF .NO_MAPS_INUSE EQL 0
: 2121 6      THEN $DS_ABORT(ARG = TEST);
: 2122 6      END;
: 2123 5
: 2124 5      INCR N FROM 0 TO 3 DO
: 2125 5      BEGIN
: 2126 6      CODECOMMENT
: 2127 6      'Next, set up the buffers. The parameters are buffer number',
: 2128 6      'and the pattern to use.',
: 2129 6      '':
: 2130 6      BEGIN
: 2131 7      BUFFER_SETUP(0,.WORD_PATTERN_0[.N],1); ! M10
: 2132 7      BUFFER_SETUP(1,.WORD_PATTERN_1[.N],1); ! M10
: 2133 7      BUFFER_SETUP(2,.WORD_PATTERN_2[.N],1); ! M10
: 2134 7      UBE_INIT_DATA[0] = .WORD_PATTERN_0[.N]; ! A10
: 2135 7      UBE_INIT_DATA[1] = .WORD_PATTERN_1[.N]; ! A10
: 2136 7      UBE_INIT_DATA[2] = .WORD_PATTERN_2[.N]; ! A10
: 2137 7      END;
: 2138 6
: 2139 6      CODECOMMENT
: 2140 6      'Set up the UBEs and UBI for the transfer. The parameters',
: 2141 6      'are UBE number, starting map number, offset mode, data path',
: 2142 6      'number, rotate data path mode, and data pattern if DATIP.',
: 2143 6      '':
: 2144 6      BEGIN
: 2145 7      XFER_SETUP(0,.UBE_MAP_NO[0],.UBE_DP_NO[0],0); ! M10
: 2146 7
: 2147 7
: 2148 7
: 2149 7

```



```

: 2150 7          XFER_SETUP(1,.UBE_MAP_NO[1],.UBE_DP_NO[1],0);      ! M10
: 2151 7          XFER_SETUP(2,.UBE_MAP_NO[2],.UBE_DP_NO[2],0);      ! M10
: 2152 6          END;
: 2153 6
: 2154 6          CODECOMMENT
: 2155 6          ''
: 2156 6          'Set up the UBE data buffer and the expected data for the',
: 2157 6          'transfer analysis routine. Set the IPL to 16 to allow the',
: 2158 6          'UBE to interrupt on error. Then start the transfer.',
: 2159 6          '';
: 2160 7          BEGIN
: 2161 7          $DS_BREAK;
: 2162 7          INCR X FROM 0 TO 2 DO                                  ! A8
: 2163 8              BEGIN
: 2164 8                  UBE_STAT_ERR[.X] = 0;                          ! A8
: 2165 8                  UBE_XFER_ERR[.X] = 0;                          ! A8
: 2166 7              END;
: 2167 7          UBE_DB(0) = NOT .WORD_PATTERN_0[.N];
: 2168 7          UBE_DB(1) = NOT .WORD_PATTERN_1[.N];
: 2169 7          UBE_DB(2) = NOT .WORD_PATTERN_2[.N];
: 2170 7          UBE_EXP_DATA[0] = NOT .WORD_PATTERN_0[.N];
: 2171 7          UBE_EXP_DATA[1] = NOT .WORD_PATTERN_1[.N];
: 2172 7          UBE_EXP_DATA[2] = NOT .WORD_PATTERN_2[.N];
: 2173 7          ! $DS_SETIPL(LEVEL=X'16');                             ! D8
: 2174 7          UBE_CR1(0) = DATO_BR4;
: 2175 7          UBE_CR1(1) = DATO_BR5;
: 2176 7          UBE_CR1(2) = DATO_BR6;
: 2177 6          END;
: 2178 6
: 2179 6          CODECOMMENT
: 2180 6          ''
: 2181 6          'Wait for enough time to allow the transfer to complete.',
: 2182 6          'then check to see if a status error was indicated. If it',
: 2183 6          'was, then dump the UBI and UBE registers. If there a',
: 2184 6          'status error was not indicated, then check to see if there',
: 2185 6          'was a data error. If so, report it.',
: 2186 6          '';
: 2187 7          BEGIN
: 2188 7          ! $DS_WAITMS(TIME=30);                                    ! D8
: 2189 7          UBE_INT_OCC = 0;                                         ! A8
: 2190 7          CURRENT_MAP = .NO_MAPS_INUSE - 1;                       ! A10
: 2191 7          WHILE 1 DO                                              ! A8
: 2192 8              BEGIN
: 2193 9                  IF (.UBE_INT_OCC[0] AND .UBE_INT_OCC[1] AND .UBE_INT_OCC[2]) ! A8
: 2194 8                      THEN EXITLOOP;                             ! A8
: 2195 8                      IF .CURRENT_MAP EQL 511                    ! A10
: 2196 8                          THEN CURRENT_MAP = .NO_MAPS_INUSE      ! A10
: 2197 8                          ELSE CURRENT_MAP = .CURRENT_MAP + 1;    ! A10
: 2198 8
: 2199 8                      !
: 2200 8                      ! Write and Read the Map Register
: 2201 8                      !
: 2202 8                      UBI_MAP(.CURRENT_MAP) = 0;                 ! A10
: 2203 8                      IF (MAP_DATA = (.UBI_MAP(.CURRENT_MAP) AND MAP_MASK)) NEQ 0 ! A10
: 2204 8                      THEN

```

```

; 2205 9          BEGIN
; P 2206 9          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,PRLINK=PRINT_MAP_ERR,! A10 M13
; P 2207 9          P1=.CURRENT_MAP,P2=UBI_MAP(.CURRENT_MAP),
; L 2208 9          P3=FMT_MAP_FAIL,P4=0,P5=.MAP_DATA);
; xPRINT:          Test 32, Subtest 0, Error 12
; 2209 9          ! $DS_PRINTX(FMT_MAP_FAIL1,.CURRENT_MAP,0,.MAP_DATA);! A10 D13
; 2210 9          END
; 2211 8          ELSE
; 2212 9          BEGIN
; 2213 9          UBI_MAP(.CURRENT_MAP) = -1; ! A10
; 2214 9          IF (MAP_DATA = (.UBI_MAP(.CURRENT_MAP) AND MAP_MASK)) NEQ MAP_MASK ! A10
; 2215 9          THEN ! A10
; 2216 10          BEGIN
; P 2217 10          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,PRLINK=PRINT_MAP_ERR,! A10 M13
; P 2218 10          P1=.CURRENT_MAP,P2=UBI_MAP(.CURRENT_MAP),
; L 2219 10          P3=FMT_MAP_FAIL,P4=MAP_MASK,P5=.MAP_DATA);
; xPRINT:          Test 32, Subtest 0, Error 13
; 2220 10          ! $DS_PRINTX(FMT_MAP_FAIL1,.CURRENT_MAP,MAP_MASK,.MAP_DATA);! A10 D13
; 2221 9          END; ! A10
; 2222 8          END; ! A10
; 2223 7          END;
; 2224 7          IF .UBE_STAT_ERR[0]
; 2225 7          THEN
; 2226 8          BEGIN
; L 2227 8          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP); ! M7
; xPRINT:          Test 32, Subtest 0, Error 14
; 2228 8          UBI_DUMP;
; 2229 8          UBE_DUMP(0);
; 2230 8          END
; 2231 7          ELSE
; 2232 8          BEGIN
; 2233 8          IF .UBE_XFER_ERR[0]
; 2234 8          THEN
; P 2235 8          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP, ! M7
; L 2236 8          PRLINK=PRINT_DCMP_ERR,P1=0);
; xPRINT:          Test 32, Subtest 0, Error 15
; 2237 7          END;
; 2238 7          IF .UBE_STAT_ERR[1]
; 2239 7          THEN
; 2240 8          BEGIN
; L 2241 8          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP); ! M7
; xPRINT:          Test 32, Subtest 0, Error 16
; 2242 8          UBI_DUMP;
; 2243 8          UBE_DUMP(1);
; 2244 8          END
; 2245 7          ELSE
; 2246 8          BEGIN
; 2247 8          IF .UBE_XFER_ERR[1]
; 2248 8          THEN
; P 2249 8          $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP, ! M7
; L 2250 8          PRLINK=PRINT_DCMP_ERR,P1=1);
; xPRINT:          Test 32, Subtest 0, Error 17
; 2251 7          END;
; 2252 7          IF .UBE_STAT_ERR[2]
; 2253 7          THEN

```

```

; 2254 8
; L 2255 8
; xPRINT: Test 32, Subtest 0, Error 18
; 2256 8
; 2257 8
; 2258 8
; 2259 7
; 2260 8
; 2261 8
; 2262 8
; P 2263 8
; L 2264 8
; xPRINT: Test 32, Subtest 0, Error 19
; 2265 7
; 2266 6
; 2267 5
; 2268 5
; 2269 4
; 2270 3
; 2271 3
; 2272 4
; 2273 4
; 2274 4
; 2275 5
; 2276 5
; 2277 5
; 2278 5
; 2279 5
; 2280 5
; 2281 5
; 2282 5
; 2283 5
; 2284 5
; 2285 5
; 2286 5
; 2287 5
; 2288 5
; 2289 5
; 2290 5
; 2291 5
; 2292 5
; 2293 5
; 2294 6
; 2295 6
; 2296 6
; 2297 6
; 2298 5
; 2299 5
; 2300 5
; 2301 5
; 2302 5
; 2303 5
; 2304 5
; 2305 5
; 2306 6

      BEGIN
      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP);      ! M7
      UBI_DUMP;
      UBE_DUMP(2);
      END
    ELSE
      BEGIN
      IF .UBE_XFER_ERR[2]
      THEN
        $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,      ! M7
          PRLINK=PRINT_DCMP_ERR,P1=2);
      END;
      END;
      $DS_RELBUF(PAGCNT=.UBE_PAGE_CNT);
      END;
      END;
      ! INCR N
      ! A8
      ! INCR I
      ! CASE
[4]: BEGIN
      INCR I FROM 0 TO 3 DO      ! Data path select      ! A10
      BEGIN
      UBE_DP_NO[0] = .I;      ! A10
      IF .UBE_DP_NO[0] EQL 3      ! A10
      THEN UBE_DP_NO[1] = 0      ! A10
      ELSE UBE_DP_NO[1] = .UBE_DP_NO[0] + 1;      ! A10
      IF .UBE_DP_NO[1] EQL 3      ! A10
      THEN UBE_DP_NO[2] = 0      ! A10
      ELSE UBE_DP_NO[2] = .UBE_DP_NO[1] + 1;      ! A10
      IF .UBE_DP_NO[2] EQL 3      ! A10
      THEN UBE_DP_NO[3] = 0      ! A10
      ELSE UBE_DP_NO[3] = .UBE_DP_NO[2] + 1;      ! A10
      UBE_MODE[0] = 0;      ! A10
      UBE_MODE[1] = 1;      ! A10
      UBE_MODE[2] = 0;      ! A10
      UBE_MODE[3] = 1;      ! A10
      BEGIN
      LOCAL X;
      INCR X FROM 0 TO 3 DO      ! A8
      UBE_CE(.X) = 0;      ! A8
      END;
      CODECOMMENT
      'This is the action taken for four UBEs. First, the map',
      'buffers routine is called. This routine allocates buffer space',
      'for the transfers.',
      '':
      BEGIN

```



```

: 2307 6      NO_MAPS_INUSE = MAP_BUFFERS(4,1);      ! M10
: 2308 6      IF .NO_MAPS_INUSE EQL 0
: 2309 6      THEN $DS_ABORT(ARG = TEST);
: 2310 5      END;
: 2311 5
: 2312 5      INCR N FROM 0 TO 3 DO
: 2313 6      BEGIN
: 2314 6
: 2315 6      CODECOMMENT
: 2316 6      ..
: 2317 6      'Next, set up the buffers. The parameters are buffer number',
: 2318 6      'and the pattern to use.',
: 2319 6      ..
: 2320 7      BEGIN
: 2321 7      BUFFER_SETUP(0,.WORD_PATTERN_0[.N],1);      ! M10
: 2322 7      BUFFER_SETUP(1,.WORD_PATTERN_1[.N],1);      ! M10
: 2323 7      BUFFER_SETUP(2,.WORD_PATTERN_2[.N],1);      ! M10
: 2324 7      BUFFER_SETUP(3,.WORD_PATTERN_3[.N],1);      ! M10
: 2325 7      UBE_INIT_DATA[0] = .WORD_PATTERN_0[.N];      ! M10
: 2326 7      UBE_INIT_DATA[1] = .WORD_PATTERN_1[.N];      ! M10
: 2327 7      UBE_INIT_DATA[2] = .WORD_PATTERN_2[.N];      ! M10
: 2328 7      UBE_INIT_DATA[3] = .WORD_PATTERN_3[.N];      ! M10
: 2329 6      END;
: 2330 6
: 2331 6      CODECOMMENT
: 2332 6      ..
: 2333 6      'Set up the UBEs and UBI for the transfer. The parameters',
: 2334 6      'are UBE number, starting map number, offset mode, data path',
: 2335 6      'number, rotate data path mode, and data pattern if DATIP.',
: 2336 6      ..
: 2337 7      BEGIN
: 2338 7      XFER_SETUP(0,.UBE_MAP_NO[0],.UBE_DP_NO[0],0);      ! M10
: 2339 7      XFER_SETUP(1,.UBE_MAP_NO[1],.UBE_DP_NO[1],0);      ! M10
: 2340 7      XFER_SETUP(2,.UBE_MAP_NO[2],.UBE_DP_NO[2],0);      ! M10
: 2341 7      XFER_SETUP(3,.UBE_MAP_NO[3],.UBE_DP_NO[3],0);      ! M10
: 2342 6      END;
: 2343 6
: 2344 6      CODECOMMENT
: 2345 6      ..
: 2346 6      'Set up the UBE data buffer and the expected data for the',
: 2347 6      'transfer analysis routine. Set the IPL to 16 to allow the',
: 2348 6      'UBE to interrupt on error. Then start the transfer.',
: 2349 6      ..
: 2350 7      BEGIN
: 2351 7      $DS_BREAK;
: 2352 7      INCR X FROM 0 TO 3 DO      ! A8
: 2353 8      BEGIN
: 2354 8      UBE_STAT_ERR[.X] = 0;      ! A8
: 2355 8      UBE_XFER_ERR[.X] = 0;      ! A8
: 2356 7      END;
: 2357 7      UBE_DB(0) = NOT .WORD_PATTERN_0[.N];
: 2358 7      UBE_DB(1) = NOT .WORD_PATTERN_1[.N];
: 2359 7      UBE_DB(2) = NOT .WORD_PATTERN_2[.N];
: 2360 7      UBE_DB(3) = NOT .WORD_PATTERN_3[.N];
: 2361 7      UBE_EXP_DATA[0] = NOT .WORD_PATTERN_0[.N];

```

```

: 2362 7      UBE_EXP_DATA[1] = NOT .WORD_PATTERN_1[.N];
: 2363 7      UBE_EXP_DATA[2] = NOT .WORD_PATTERN_2[.N];
: 2364 7      UBE_EXP_DATA[3] = NOT .WORD_PATTERN_3[.N];
: 2365 7      ! $DS_SETIPL(LEVEL=X'16'); ! D8
: 2366 7      UBE_CR1(0) = DATO_BR4;
: 2367 7      UBE_CR1(1) = DATO_BR5;
: 2368 7      UBE_CR1(2) = DATO_BR6;
: 2369 7      UBE_CR1(3) = DATO_BR7;
: 2370 6      END;
: 2371 6
: 2372 6      CODECOMMENT
: 2373 6      'Wait for enough time to allow the transfer to complete.',
: 2374 6      'then check to see if a status error was indicated. If it',
: 2375 6      'was, then dump the UBI and UBE registers. If there a',
: 2376 6      'status error was not indicated, then check to see if there',
: 2377 6      'was a data error. If so, report it.',
: 2378 6      ' ';
: 2379 6
: 2380 7      BEGIN
: 2381 7      ! $DS_WAITMS(TIME=100); ! D8
: 2382 7      UBE_INT_OCC = 0; ! A8
: 2383 7      CURRENT_MAP = .NO_MAPS_INUSE - 1; ! A10
: 2384 7      WHILE 1 DO ! A8
: 2385 8      BEGIN
: 2386 9      IF (.UBE_INT_OCC[0] AND .UBE_INT_OCC[1] ! A8
: 2387 9      AND .UBE_INT_OCC[2] AND .UBE_INT_OCC[3]) ! A8
: 2388 8      THEN EXITLOOP; ! A8
: 2389 8      IF .CURRENT_MAP EQL 511 ! A10
: 2390 8      THEN CURRENT_MAP = .NO_MAPS_INUSE ! A10
: 2391 8      ELSE CURRENT_MAP = .CURRENT_MAP + 1; ! A10
: 2392 8
: 2393 8      !
: 2394 8      ! Write and Read the Map Register
: 2395 8      !
: 2396 8      UBI_MAP(.CURRENT_MAP) = 0; ! A10
: 2397 8      IF (MAP_DATA = (.UBI_MAP(.CURRENT_MAP) AND MAP_MASK)) NEQ 0 ! A10
: 2398 8      THEN
: 2399 9      BEGIN
: 2400 9      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,PRLINK=PRINT_MAP_ERR,! A10 M13
: 2401 9      P1=.CURRENT_MAP,P2=UBI_MAP(.CURRENT_MAP),
: 2402 9      P3=FMT_MAP_FAIL,P4=0,P5=.MAP_DATA);
: 2403 9      Test 32, Subtest 0, Error 20
: 2404 9      ! $DS_PRINTX(FMT_MAP_FAIL1,.CURRENT_MAP,0,.MAP_DATA);! A10 D13
: 2405 8      END
: 2406 9      ELSE
: 2407 9      BEGIN
: 2408 9      UBI_MAP(.CURRENT_MAP) = -1; ! A10
: 2409 9      IF (MAP_DATA = (.UBI_MAP(.CURRENT_MAP) AND MAP_MASK)) NEQ MAP_MASK ! A10
: 2410 10      THEN ! A10
: 2411 10      BEGIN
: 2412 10      $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,PRLINK=PRINT_MAP_ERR,! A10 M13
: 2413 10      P1=.CURRENT_MAP,P2=UBI_MAP(.CURRENT_MAP),
: 2414 10      P3=FMT_MAP_FAIL,P4=MAP_MASK,P5=.MAP_DATA);
: 2414 10      Test 32, Subtest 0, Error 21
: 2414 10      ! $DS_PRINTX(FMT_MAP_FAIL1,.CURRENT_MAP,MAP_MASK,.MAP_DATA);! A10 D13

```

```

: 2415 9                                END;                                ! A10
: 2416 8                                END;                                ! A10
: 2417 7                                END;
: 2418 7                                IF .UBE_STAT_ERR[0]
: 2419 7                                THEN
: 2420 8                                BEGIN
: L 2421 8                                $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP);    ! M7
: xPRINT:                                Test 32, Subtest 0, Error 22
: 2422 8                                UBI_DUMP;
: 2423 8                                UBE_DUMP(0);
: 2424 8                                END
: 2425 7                                ELSE
: 2426 8                                BEGIN
: 2427 8                                IF .UBE_XFER_ERR[0]
: 2428 8                                THEN
: P 2429 8                                $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,    ! M7
: L 2430 8                                PRLINK=PRINT_DCMP_ERR,P1=0);
: xPRINT:                                Test 32, Subtest 0, Error 23
: 2431 7                                END;
: 2432 7                                IF .UBE_STAT_ERR[1]
: 2433 7                                THEN
: 2434 8                                BEGIN
: L 2435 8                                $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP);    ! M7
: xPRINT:                                Test 32, Subtest 0, Error 24
: 2436 8                                UBI_DUMP;
: 2437 8                                UBE_DUMP(1);
: 2438 8                                END
: 2439 7                                ELSE
: 2440 8                                BEGIN
: 2441 8                                IF .UBE_XFER_ERR[1]
: 2442 8                                THEN
: P 2443 8                                $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,    ! M7
: L 2444 8                                PRLINK=PRINT_DCMP_ERR,P1=1);
: xPRINT:                                Test 32, Subtest 0, Error 25
: 2445 7                                END;
: 2446 7                                IF .UBE_STAT_ERR[2]
: 2447 7                                THEN
: 2448 8                                BEGIN
: L 2449 8                                $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP);    ! M7
: xPRINT:                                Test 32, Subtest 0, Error 26
: 2450 8                                UBI_DUMP;
: 2451 8                                UBE_DUMP(2);
: 2452 8                                END
: 2453 7                                ELSE
: 2454 8                                BEGIN
: 2455 8                                IF .UBE_XFER_ERR[2]
: 2456 8                                THEN
: P 2457 8                                $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP,    ! M7
: L 2458 8                                PRLINK=PRINT_DCMP_ERR,P1=2);
: xPRINT:                                Test 32, Subtest 0, Error 27
: 2459 7                                END;
: 2460 7                                IF .UBE_STAT_ERR[3]
: 2461 7                                THEN
: 2462 8                                BEGIN
: L 2463 8                                $DS_ERRHARD(UNIT=.LUN,MSGADR=UBIMOD_BDP);    ! M7

```


Test 32, Subtest 0, Error 29

														.PSECT DISPATCH,NOWRT,NOEXE,2					
00000000' 00000000' 00000000' 00000000' 00078 \$:														.ADDRESS P.AAP, P.AAQ, P.AAR, TEST_032		:			
00000000 00000007 00088														.LONG 7, 0		:			
														.PSECT \$PLIT\$,NOWRT,NOEXE,2					
6E	61	72	54	20	6B	63	6F	6C	42	20	45	42	55	0020	00090	P.AAP:	.WORD 32	:	
						74	73	65	54	20	72	65	66	17	00092	P.AAQ:	.ASCII <23>\UBE Block Transfer Test\	:	
														73	000A1		:		
															000AA		:		
													00000000	000AC	P.AAR:	.BLKB 2	:		
																.LONG 0	:		
														.EXTRN DS\$PRINTX, DS\$RELBUF					
														.PSECT TEST_032,NOWRT,2					
														.ENTRY TEST_032, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1666		:			
																	R11	:	
						5E		BC	AE	9E	00002						MOVAB	-68(SP), SP	:
						53		0000G	CF	D0	00006						MOVL	NUM_UBE, R3	: 1748
									16	12	0000B						BNEQ	1\$	:
						02		0000FE54	9F	D1	0000D						CMPL	a#^X0000FE54, #2	:
									0D	18	00014						BGEQ	1\$	:
									CF	9F	00016						PUSHAB	FMT_NO_UBE	: 1749
						00000000G			01	FB	0001A						CALLS	#1, a#DS\$PRINTX	:
									3D	11	00021						BRB	4\$	:

Address	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
---------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

```
;
; Next, set up the buffer.  The parameters are buffer number
; and the pattern to use.
```

7E	0000G	CF	9A	000D7	9\$:	MOVZBL	UBE_MODE, -(SP)	:	1831
7E	40	AE42	3C	000DC		MOVZWL	WORD_PATTERN_0[N], -(SP)	:	
		7E	D4	000E1		CLRL	-(SP)	:	
0000G	CF	03	FB	000E3		CALLS	#3, BUFFER_SETUP	:	
0000G	CF	3C	AE42	B0	000E8	MOVW	WORD_PATTERN_0[N], UBE_INIT_DATA	:	1832

```
; Set up the UBE and UBI for the transfer.  The parameters
; are UBE number, starting map number, offset mode, data path
; number, rotate data path mode.
```

Address	Op Code	Op Name	Comments	PC
0000G	CF 95	000EF	TSTB	UBE_MODE ; 1842
	17 12	000F3	BNEQ	10\$;
	01 DD	000F5	PUSHL	#1 ; 1845
	7E D4	000F7	CLRL	-(SP) ;
7E 0000G	CF 3C	000F9	MOVZWL	UBE_MAP_NO, -(SP) ;
	7E D4	000FE	CLRL	-(SP) ;
0000G CF	04 FB	00100	CALLS	#4, XFER_SETUP ;
0000G CF	01 8E	00105	MNEGB	#1, UBE_DP_NO ; 1846
	12 11	0010A	BRB	11\$; 1842
	7E 7C	0010C	CLRQ	-(SP) ; 1850
7E 0000G	CF 3C	0010E	MOVZWL	UBE_MAP_NO, -(SP) ;
	7E D4	00113	CLRL	-(SP) ;
0000G CF	04 FB	00115	CALLS	#4, XFER_SETUP ;
0000G	CF 94	0011A	CLRB	UBE_DP_NO ; 1851

```
; Set up the UBE data buffer and the expected data for the
; transfer analysis routine. Set the IPL to 16 to allow the
; UBE to interrupt on error. Then start the transfer.
```

Address	Op Code	Op Name	Comment	PC
00000000G	9F	00 FB 0011E	CALLS #0, @DS\$BREAK	1861
0000G	CF	01 8A 00125	BICB2 #1, UBE_STAT_ERR	1863
0000G	CF	01 8A 0012A	BICB2 #1, UBE_XFER_ERR	1864
	50	0000G CF D0 0012F	MOVL UBE_BASE_ADDR, R0	1865
	51	3C AE42 3C 00134	MOVZWL WORD_PATTERN_0[N], R1	
	51	51 D2 00139	MCOML R1, R1	
	60	51 B0 0013C	MOVW R1, (R0)	
0000G	CF	51 B0 0013F	MOVW R1, UBE_EXP_DATA	1866
06	A0	0663 8F B0 00144	MOVW #1635, 6(R0)	1868

```
; Wait for the transfer to complete,
; then check to see if a status error was indicated.  If it
; was, then dump the UBI and UBE registers.  If there a
; status error was not indicated, then check to see if there
; was a data error.  If so, report it.
```

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

	53	1C	AE	D0	00164	MOVL	NO_MAPS_INUSE, CURRENT_MAP	:	1888
			02	11	00168	BRB	15\$	:	
			53	D6	0016A	INCL	CURRENT_MAP	:	1889
	50	0000GDF	43	DE	0016C	MOVAL	@UBI_UNDER_TEST[CURRENT_MAP], R0	:	1894
	54	0800	C0	9E	00172	MOVAB	2048(R0), R4	:	
			64	D4	00177	CLRL	(R4)	:	
57	64	7D9F8000	8F	CB	00179	BICL3	#2107604992, (R4), MAP_DATA	:	1895
			1D	13	00181	BEQL	16\$	:	
			7E	D4	00183	CLRL	-(SP)	:	1900
			57	DD	00185	PUSHL	MAP_DATA	:	
			7E	D4	00187	CLRL	-(SP)	:	
		0000G	CF	9F	00189	PUSHAB	FMT_MAP_FAIL	:	
			18	BB	0018D	PUSHR	#^M<R3,R4>	:	
		0000G	CF	9F	0018F	PUSHAB	PRINT_MAP_ERR	:	
		0000G	CF	9F	00193	PUSHAB	UBIMOD_BDP	:	
	7E	0000G	CF	9A	00197	MOVZBL	LUN, -(SP)	:	
			02	DD	0019C	PUSHL	#2	:	
			33	11	0019E	BRB	17\$	:	
	64		01	CE	001A0	MNEGL	#1, (R4)	:	1905
57	64	7D9F8000	8F	CB	001A3	BICL3	#2107604992, (R4), MAP_DATA	:	1906
82607FFF	8F		57	D1	001AB	CMPL	MAP_DATA, #-2107604993	:	
			9F	13	001B2	BEQL	12\$	:	
			7E	D4	001B4	CLRL	-(SP)	:	1911
			57	DD	001B6	PUSHL	MAP_DATA	:	
		82607FFF	8F	DD	001B8	PUSHL	#-2107604993	:	
		0000G	CF	9F	001BE	PUSHAB	FMT_MAP_FAIL	:	
			18	BB	001C2	PUSHR	#^M<R3,R4>	:	
		0000G	CF	9F	001C4	PUSHAB	PRINT_MAP_ERR	:	
		0000G	CF	9F	001C8	PUSHAB	UBIMOD_BDP	:	
	7E	0000G	CF	9A	001CC	MOVZBL	LUN, -(SP)	:	
			03	DD	001D1	PUSHL	#3	:	
00000000G	9F		0A	FB	001D3	CALLS	#10, @DS\$ERRHARD	:	
			FF76	31	001DA	BRW	12\$	:	1883
	03	0000G	CF	E8	001DD	BLBS	UBE_STAT_ERR, 19\$	:	1916
			0094	31	001E2	BRW	20\$	:	
			7E	7C	001E5	CLRQ	-(SP)	:	1919
			7E	7C	001E7	CLRQ	-(SP)	:	
			7E	7C	001E9	CLRQ	-(SP)	:	
			7E	D4	001EB	CLRL	-(SP)	:	
		0000G	CF	9F	001ED	PUSHAB	UBIMOD_BDP	:	
	7E	0000G	CF	9A	001F1	MOVZBL	LUN, -(SP)	:	
			04	DD	001F6	PUSHL	#4	:	
00000000G	9F		0A	FB	001F8	CALLS	#10, @DS\$ERRHARD	:	
	50	0000G	CF	D0	001FF	MOVL	UBI_UNDER_TEST, R0	:	
55	60	FFFE	8F	AB	00204	BICW3	#65534, (R0), TEMP4	:	
5B	04	A0	1FFFFFFE	8F	CB	0020A	BICL3	#536870910, 4(R0), TEMP5	:
5A	08	A0	1FFFFFFE	8F	CB	00213	BICL3	#536870910, 8(R0), TEMP6	:
59	0C	A0	1FFFFFFE	8F	CB	0021C	BICL3	#536870910, 12(R0), TEMP7	:
			59	DD	00225	PUSHL	TEMP7	:	
			5A	DD	00227	PUSHL	TEMP6	:	
			5B	DD	00229	PUSHL	TEMP5	:	
	7E		55	3C	0022B	MOVZWL	TEMP4, -(SP)	:	
		0000G	CF	9F	0022E	PUSHAB	FMT_UBI_DUMP	:	
00000000G	9F		05	FB	00232	CALLS	#5, @DS\$PRINTX	:	
	50	0000G	CF	D0	00239	MOVL	UBE_BASE_ADDR, R0	:	1921

Address	Op Code	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	---------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

```

52 D4 002FF 26$: CLRL N ; 1963
;
; Next, set up the buffer. The parameters are buffer number
; and the pattern to use.
;
7E D4 00301 27$: CLRL -(SP) ; 1972
7E 40 AE42 3C 00303 MOVZWL WORD_PATTERN_0[N], -(SP) ;
7E D4 00308 CLRL -(SP) ;
0000G CF 03 FB 0030A CALLS #3, BUFFER_SETUP ;
01 DD 0030F PUSHL #1 ; 1973
7E 38 AE42 3C 00311 MOVZWL WORD_PATTERN_1[N], -(SP) ;
01 DD 00316 PUSHL #1 ;
0000G CF 03 FB 00318 CALLS #3, BUFFER_SETUP ;
0000G CF 3C AE42 B0 0031D MOVW WORD_PATTERN_0[N], UBE_INIT_DATA ; 1974
0000G CF 34 AE42 B0 00324 MOVW WORD_PATTERN_1[N], UBE_INIT_DATA+2 ; 1975
;
; Set up the UBEs and UBI for the transfer. The parameters
; are UBE number, starting map number, offset mode, data path
; number, rotate data path mode, and data pattern if DATIP.
;
7E D4 0032B CLRL -(SP) ; 1985
7E 0000G CF 9A 0032D MOVZBL UBE_DP_NO, -(SP) ;
7E 0000G CF 3C 00332 MOVZWL UBE_MAP_NO, -(SP) ;
0000G CF 7E D4 00337 CLRL -(SP) ;
04 FB 00339 CALLS #4, XFER_SETUP ;
7E D4 0033E CLRL -(SP) ; 1986
7E 0000G CF 9A 00340 MOVZBL UBE_DP_NO+1, -(SP) ;
7E 0000G CF 3C 00345 MOVZWL UBE_MAP_NO+2, -(SP) ;
01 DD 0034A PUSHL #1 ;
0000G CF 04 FB 0034C CALLS #4, XFER_SETUP ;
;
; Set up the UBE data buffer and the expected data for the
; transfer analysis routine. Set the IPL to 16 to allow the
; UBE to interrupt on error. Then start the transfer.
;
00000000G 9F 00 FB 00351 CALLS #0, @DS$BREAK ; 1996
50 D4 00358 CLRL X ; 1998
00 0000G CF 50 E5 0035A 28$: BBCC X, UBE_STAT_ERR, 29$ ; 2000
00 0000G CF 50 E5 00360 29$: BBCC X, UBE_XFER_ERR, 30$ ; 2001
F0 50 01 F3 00366 30$: AOBLEQ #1, X, 28$ ; 1998
51 0000G CF D0 0036A MOVL UBE_BASE_ADDR, R1 ; 2003
56 3C AE42 3C 0036F MOVZWL WORD_PATTERN_0[N], R6 ;
56 56 D2 00374 MCOML R6, R6 ;
61 56 B0 00377 MOVW R6, (R1) ;
50 0000G CF D0 0037A MOVL UBE_BASE_ADDR+4, R0 ; 2004
54 34 AE42 3C 0037F MOVZWL WORD_PATTERN_1[N], R4 ;
54 54 D2 00384 MCOML R4, R4 ;
60 54 B0 00387 MOVW R4, (R0) ;
0000G CF 56 B0 0038A MOVW R6, UBE_EXP_DATA ; 2005
0000G CF 54 B0 0038F MOVW R4, UBE_EXP_DATA+2 ; 2006
06 A1 0663 8F B0 00394 MOVW #1635, 6(R1) ; 2008
06 A0 0665 8F B0 0039A MOVW #1637, 6(R0) ; 2009
;
; Wait for enough time to allow the transfer to complete,
; then check to see if a status error was indicated. If it

```


; was, then dump the UBI and UBE registers. If there a
; status error was not indicated, then check to see if there
; was a data error. If so, report it.
;

53	1C	AE	0000G	CF	D4	003A0	CLRL	UBE_INT_OCC	; 2022
		08		01	C3	003A4	SUBL3	#1, NO MAPS_INUSE, CURRENT_MAP	; 2023
		03	0000G	CF	E9	003A9	BLBC	UBE_INT_OCC, 32\$	; 2026
			0000G	CF	E9	003AE	BLBC	UBE_INT_OCC+1, 32\$	; :
				0082	31	003B3	BRW	37\$	; :
	000001FF	8F		53	D1	003B6	CMPL	CURRENT_MAP, #511	; 2028
				06	12	003BD	BNEQ	33\$	; :
		53	1C	AE	D0	003BF	MOVL	NO MAPS_INUSE, CURRENT_MAP	; 2029
				02	11	003C3	BRB	34\$	; :
				53	D6	003C5	INCL	CURRENT_MAP	; 2030
		50	0000G	DF	43	DE	003C7	MOVAL	@UBI_UNDER_TEST[CURRENT_MAP], R0
		54	0800	C0	9E	003CD	MOVAB	2048(R0), R4	; 2035
				64	D4	003D2	CLRL	(R4)	; :
57	64	7D9F8000		8F	CB	003D4	BICL3	#2107604992, (R4), MAP_DATA	; 2036
				1D	13	003DC	BEQL	35\$	; :
				7E	D4	003DE	CLRL	-(SP)	; 2041
				57	DD	003E0	PUSHL	MAP_DATA	; :
				7E	D4	003E2	CLRL	-(SP)	; :
			0000G	CF	9F	003E4	PUSHAB	FMT_MAP_FAIL	; :
				18	BB	003E8	PUSHR	#^M<R3,R4>	; :
			0000G	CF	9F	003EA	PUSHAB	PRINT_MAP_ERR	; :
			0000G	CF	9F	003EE	PUSHAB	UBIMOD_BDP	; :
	7E		0000G	CF	9A	003F2	MOVZBL	LUN, -(SP)	; :
				06	DD	003F7	PUSHL	#6	; :
				33	11	003F9	BRB	36\$	; :
		64		01	CE	003FB	MNEGL	#1, (R4)	; 2046
57	64	7D9F8000		8F	CB	003FE	BICL3	#2107604992, (R4), MAP_DATA	; 2047
	82607FFF	8F		57	D1	00406	CMPL	MAP_DATA, #-2107604993	; :
				9A	13	0040D	BEQL	31\$	; :
				7E	D4	0040F	CLRL	-(SP)	; 2052
				57	DD	00411	PUSHL	MAP_DATA	; :
		82607FFF		8F	DD	00413	PUSHL	#-2107604993	; :
			0000G	CF	9F	00419	PUSHAB	FMT_MAP_FAIL	; :
				18	BB	0041D	PUSHR	#^M<R3,R4>	; :
			0000G	CF	9F	0041F	PUSHAB	PRINT_MAP_ERR	; :
			0000G	CF	9F	00423	PUSHAB	UBIMOD_BDP	; :
	7E		0000G	CF	9A	00427	MOVZBL	LUN, -(SP)	; :
				07	DD	0042C	PUSHL	#7	; :
		00000000G	9F	0A	FB	0042E	CALLS	#10, @DS\$ERRHARD	; :
				FF71	31	00435	BRW	31\$	; 2024
		03	0000G	CF	E8	00438	BLBS	UBE_STAT_ERR, 38\$	; 2057
				0094	31	0043D	BRW	39\$	; :
				7E	7C	00440	CLRQ	-(SP)	; 2060
				7E	7C	00442	CLRQ	-(SP)	; :
				7E	7C	00444	CLRQ	-(SP)	; :
				7E	D4	00446	CLRL	-(SP)	; :
			0000G	CF	9F	00448	PUSHAB	UBIMOD_BDP	; :
	7E		0000G	CF	9A	0044C	MOVZBL	LUN, -(SP)	; :
				08	DD	00451	PUSHL	#8	; :
		00000000G	9F	0A	FB	00453	CALLS	#10, @DS\$ERRHARD	; :
	50		0000G	CF	D0	0045A	MOVL	UBI_UNDER_TEST, R0	; :

55		60	FFFE	8F	AB	0045F	BICW3	#65534, (R0), TEMP4	:
5B	04	A0	1FFFFFFE	8F	CB	00465	BICL3	#536870910, 4(R0), TEMP5	:
5A	08	A0	1FFFFFFE	8F	CB	0046E	BICL3	#536870910, 8(R0), TEMP6	:
59	0C	A0	1FFFFFFE	8F	CB	00477	BICL3	#536870910, 12(R0), TEMP7	:
				59	DD	00480	PUSHL	TEMP7	:
				5A	DD	00482	PUSHL	TEMP6	:
				5B	DD	00484	PUSHL	TEMP5	:
		7E		55	3C	00486	MOVZWL	TEMP4, -(SP)	:
00000000G		9F	0000G	CF	9F	00489	PUSHAB	FMT_UBI_DUMP	:
		50	0000G	05	FB	0048D	CALLS	#5, a#DS\$PRINTX	:
	0C	AE		CF	D0	00494	MOVL	UBE_BASE_ADDR, R0	: 2062
	08	AE	02	60	B0	00499	MOVW	(R0), TEMP0	:
	04	AE	04	A0	B0	0049D	MOVW	2(R0), TEMP1	:
		6E	06	A0	B0	004A2	MOVW	4(R0), TEMP2	:
		55	0E	A0	B0	004A7	MOVW	6(R0), TEMP3	:
		7E		A0	B0	004AB	MOVW	14(R0), TEMP4	:
		7E		55	3C	004AF	MOVZWL	TEMP4, -(SP)	:
		7E	04	AE	3C	004B2	MOVZWL	TEMP3, -(SP)	:
		7E	0C	AE	3C	004B6	MOVZWL	TEMP2, -(SP)	:
		7E	14	AE	3C	004BA	MOVZWL	TEMP1, -(SP)	:
		7E	1C	AE	3C	004BE	MOVZWL	TEMP0, -(SP)	:
		7E	0000G	CF	9A	004C2	MOVZBL	UBE_DRIVE, -(SP)	:
			0000G	CF	9F	004C7	PUSHAB	FMT_UBE_DUMP	:
00000000G		9F		07	FB	004CB	CALLS	#7, a#DS\$PRINTX	:
		1C	0000G	21	11	004D2	BRB	40\$	: 2057
				CF	E9	004D4	BLBC	UBE_XFER_ERR, 40\$	: 2066
				7E	7C	004D9	CLRQ	-(SP)	: 2069
				7E	7C	004DB	CLRQ	-(SP)	:
				7E	7C	004DD	CLRQ	-(SP)	:
			0000G	CF	9F	004DF	PUSHAB	PRINT_DCMP_ERR	:
			0000G	CF	9F	004E3	PUSHAB	UBIMOD_BDP	:
		7E	0000G	CF	9A	004E7	MOVZBL	LUN, -(SP)	:
				09	DD	004EC	PUSHL	#9	:
00000000G		9F		0A	FB	004EE	CALLS	#10, a#DS\$ERRHARD	:
03	0000G	CF		01	E0	004F5	BBS	#1, UBE_STAT_ERR, 41\$	: 2071
				0094	31	004FB	BRW	42\$	: 2074
				7E	7C	004FE	CLRQ	-(SP)	:
				7E	7C	00500	CLRQ	-(SP)	:
				7E	7C	00502	CLRQ	-(SP)	:
				7E	D4	00504	CLRL	-(SP)	:
			0000G	CF	9F	00506	PUSHAB	UBIMOD_BDP	:
		7E	0000G	CF	9A	0050A	MOVZBL	LUN, -(SP)	:
				0A	DD	0050F	PUSHL	#10	:
00000000G		9F		0A	FB	00511	CALLS	#10, a#DS\$ERRHARD	:
		50	0000G	CF	D0	00518	MOVL	UBI_UNDER_TEST, R0	:
55		60	FFFE	8F	AB	0051D	BICW3	#65534, (R0), TEMP4	:
5B	04	A0	1FFFFFFE	8F	CB	00523	BICL3	#536870910, 4(R0), TEMP5	:
5A	08	A0	1FFFFFFE	8F	CB	0052C	BICL3	#536870910, 8(R0), TEMP6	:
59	0C	A0	1FFFFFFE	8F	CB	00535	BICL3	#536870910, 12(R0), TEMP7	:
				59	DD	0053E	PUSHL	TEMP7	:
				5A	DD	00540	PUSHL	TEMP6	:
				5B	DD	00542	PUSHL	TEMP5	:
		7E		55	3C	00544	MOVZWL	TEMP4, -(SP)	:
			0000G	CF	9F	00547	PUSHAB	FMT_UBI_DUMP	:
00000000G		9F		05	FB	0054B	CALLS	#5, a#DS\$PRINTX	:

		50	0000G	CF	D0	00552	MOVL	UBE_BASE_ADDR+4, R0	:	2076
	0C	AE		60	B0	00557	MOVW	(R0), TEMP0	:	
	08	AE	02	A0	B0	0055B	MOVW	2(R0), TEMP1	:	
	04	AE	04	A0	B0	00560	MOVW	4(R0), TEMP2	:	
		6E	06	A0	B0	00565	MOVW	6(R0), TEMP3	:	
		55	0E	A0	B0	00569	MOVW	14(R0), TEMP4	:	
		7E		55	3C	0056D	MOVZWL	TEMP4, -(SP)	:	
		7E	04	AE	3C	00570	MOVZWL	TEMP3, -(SP)	:	
		7E	0C	AE	3C	00574	MOVZWL	TEMP2, -(SP)	:	
		7E	14	AE	3C	00578	MOVZWL	TEMP1, -(SP)	:	
		7E	1C	AE	3C	0057C	MOVZWL	TEMP0, -(SP)	:	
		7E	0000G	CF	9A	00580	MOVZBL	UBE_DRIVE+1, -(SP)	:	
			0000G	CF	9F	00585	PUSHAB	FMT_UBE_DUMP	:	
	00000000G	9F		07	FB	00589	CALLS	#7, a#DS\$PRINTX	:	
				23	11	00590	BRB	43\$	:	2071
1D	0000G	CF		01	E1	00592	BBC	#1, UBE_XFER_ERR, 43\$	:	2080
				7E	7C	00598	CLRQ	-(SP)	:	2083
		7E		7E	7C	0059A	CLRQ	-(SP)	:	
				01	7D	0059C	MOVQ	#1, -(SP)	:	
			0000G	CF	9F	0059F	PUSHAB	PRINT_DCOMP_ERR	:	
			0000G	CF	9F	005A3	PUSHAB	UBIMOD_BDP	:	
		7E	0000G	CF	9A	005A7	MOVZBL	LUN, -(SP)	:	
				0B	DD	005AC	PUSHL	#11	:	
FD46	52	00000000G	9F	0A	FB	005AE	CALLS	#10, a#DS\$ERRHARD	:	
			01	03	F1	005B5	ACBL	#3, #1, N, 27\$	:	1963
				7E	7C	005BB	CLRQ	-(SP)	:	2087
			0000G	CF	DD	005BD	PUSHL	UBE_PAGE_CNT	:	
				03	FB	005C1	CALLS	#3, a#DS\$RELBUF	:	
FCEE	58	00000000G	9F	03	F1	005C8	ACBL	#3, #1, I, 23\$	:	1938
			01	09A3	31	005CE	BRW	106\$	:	1798
				20	AE	005D1	CLRL	I	:	2093
		0000G	CF	20	AE	90	MOVB	I, UBE_DP_NO	:	2095
			03	0000G	CF	91	CMPB	UBE_DP_NO, #3	:	2096
				06	12	005DF	BNEQ	46\$	:	
			0000G	CF	94	005E1	CLRB	UBE_DP_NO+1	:	2097
				08	11	005E5	BRB	47\$	:	
	0000G	CF	0000G	CF	01	81	ADDB3	#1, UBE_DP_NO, UBE_DP_NO+1	:	2098
				03	91	005EF	CMPB	UBE_DP_NO+1, #3	:	2099
				06	12	005F4	BNEQ	48\$	:	
			0000G	CF	94	005F6	CLRB	UBE_DP_NO+2	:	2100
				08	11	005FA	BRB	49\$	:	
	0000G	CF	0000G	CF	01	81	ADDB3	#1, UBE_DP_NO+1, UBE_DP_NO+2	:	2101
			0000G	CF	01	B0	MOVW	#1, UBE_MODE	:	2103
			0000G	CF	01	90	MOVB	#1, UBE_MODE+2	:	2105
				50	D4	0060E	CLRL	X	:	2109
		51	0000G	CF	40	D0	MOVL	UBE_BASE_ADDR[X], R1	:	2110
			08	A1	B4	00616	CLRW	8(R1)	:	
F3		50		02	F3	00619	AOBLEQ	#2, X, 50\$	:	

; This is the action taken for three UBES. First, the map
; buffers routine is called. This routine allocates buffer spa
ce
; for the transfers.

01 DD 0061D PUSHL #1 ; 2120


```

0000G CF      03 DD 0061F      PUSHL #3 ;
      1C AE      02 FB 00621      CALLS #2, MAP_BUFFERS ;
      50 D0 00626      MOVL R0, NO_MAPS_INUSE ;
      03 12 0062A      BNEQ 51$ ; 2121
044F 31 0062C      BRW 81$ ;
      52 D4 0062F 51$: CLRL N ; 2125
;
; Next, set up the buffers. The parameters are buffer number
; and the pattern to use.
;
      01 DD 00631 52$: PUSHL #1 ; 2134
      7E 40 AE42 3C 00633      MOVZWL WORD_PATTERN_0[N], -(SP) ;
      7E D4 00638      CLRL -(SP) ;
0000G CF      03 FB 0063A      CALLS #3, BUFFER_SETUP ;
      01 DD 0063F      PUSHL #1 ; 2135
      7E 38 AE42 3C 00641      MOVZWL WORD_PATTERN_1[N], -(SP) ;
      01 DD 00646      PUSHL #1 ;
0000G CF      03 FB 00648      CALLS #3, BUFFER_SETUP ;
      01 DD 0064D      PUSHL #1 ; 2136
      7E 30 AE42 3C 0064F      MOVZWL WORD_PATTERN_2[N], -(SP) ;
      02 DD 00654      PUSHL #2 ;
0000G CF      03 FB 00656      CALLS #3, BUFFER_SETUP ;
0000G CF      3C AE42 B0 0065B      MOVW WORD_PATTERN_0[N], UBE_INIT_DATA ; 2137
0000G CF      34 AE42 B0 00662      MOVW WORD_PATTERN_1[N], UBE_INIT_DATA+2 ; 2138
0000G CF      2C AE42 B0 00669      MOVW WORD_PATTERN_2[N], UBE_INIT_DATA+4 ; 2139
;
; Set up the UBEs and UBI for the transfer. The parameters
; are UBE number, starting map number, offset mode, data path
; number, rotate data path mode, and data pattern if DATIP.
;
      7E D4 00670      CLRL -(SP) ; 2149
      7E 0000G CF 9A 00672      MOVZBL UBE_DP_NO, -(SP) ;
      7E 0000G CF 3C 00677      MOVZWL UBE_MAP_NO, -(SP) ;
0000G CF      7E D4 0067C      CLRL -(SP) ;
      04 FB 0067E      CALLS #4, XFER_SETUP ;
      7E D4 00683      CLRL -(SP) ; 2150
      7E 0000G CF 9A 00685      MOVZBL UBE_DP_NO+1, -(SP) ;
      7E 0000G CF 3C 0068A      MOVZWL UBE_MAP_NO+2, -(SP) ;
0000G CF      01 DD 0068F      PUSHL #1 ;
      04 FB 00691      CALLS #4, XFER_SETUP ;
      7E D4 00696      CLRL -(SP) ; 2151
      7E 0000G CF 9A 00698      MOVZBL UBE_DP_NO+2, -(SP) ;
      7E 0000G CF 3C 0069D      MOVZWL UBE_MAP_NO+4, -(SP) ;
0000G CF      02 DD 006A2      PUSHL #2 ;
      04 FB 006A4      CALLS #4, XFER_SETUP ;
;
; Set up the UBE data buffer and the expected data for the
; transfer analysis routine. Set the IPL to 16 to allow the
; UBE to interrupt on error. Then start the transfer.
;
00000000G 9F      00 FB 006A9      CALLS #0, a#DS$BREAK ; 2160
      50 D4 006B0      CLRL X ; 2162
00 0000G CF      50 E5 006B2 53$: BBCC X, UBE_STAT_ERR, 54$ ; 2164
00 0000G CF      50 E5 006B8 54$: BBCC X, UBE_XFER_ERR, 55$ ; 2165
F0 50      02 F3 006BE 55$: AOBLEQ #2, X, 53$ ; 2162

```

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

```

; Wait for enough time to allow the transfer to complete,
; then check to see if a status error was indicated.  If it
; was, then dump the UBI and UBE registers.  If there a
; status error was not indicated, then check to see if there
; was a data error.  If so, report it.

```

PC	Op	OpC	OpD	OpI	OpJ	OpK	OpL	OpM	OpN	OpO	OpP	OpQ	OpR	OpS	OpT	OpU	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

			7E	D4	0078C	CLRL	-(SP)	:	2219
			57	DD	0078E	PUSHL	MAP_DATA	:	
		82607FFF	8F	DD	00790	PUSHL	#-2107604993	:	
		0000G	CF	9F	00796	PUSHAB	FMT_MAP_FAIL	:	
			18	BB	0079A	PUSHR	#AM(R3,R4)	:	
		0000G	CF	9F	0079C	PUSHAB	PRINT_MAP_ERR	:	
		0000G	CF	9F	007A0	PUSHAB	UBIMOD_BDP	:	
	7E	0000G	CF	9A	007A4	MOVZBL	LUN, -(SP)	:	
			0D	DD	007A9	PUSHL	#13	:	
	00000000G	9F	0A	FB	007AB	CALLS	#10, a#DS\$ERRHARD	:	
			FF6C	31	007B2	BRW	56\$	:	2191
	03	0000G	CF	E8	007B5	BLBS	UBE_STAT_ERR, 63\$	:	2224
		0094	31	007BA	BRW	64\$	:		
			7E	7C	007BD	CLRQ	-(SP)	:	2227
			7E	7C	007BF	CLRQ	-(SP)	:	
			7E	7C	007C1	CLRQ	-(SP)	:	
			7E	D4	007C3	CLRL	-(SP)	:	
		0000G	CF	9F	007C5	PUSHAB	UBIMOD_BDP	:	
	7E	0000G	CF	9A	007C9	MOVZBL	LUN, -(SP)	:	
			0E	DD	007CE	PUSHL	#14	:	
	00000000G	9F	0A	FB	007D0	CALLS	#10, a#DS\$ERRHARD	:	
		50	0000G	CF	D0	MOVL	UBI_UNDER_TEST, R0	:	
55		60	FFFE	8F	AB	BICW3	#65534, (R0), TEMP4	:	
5B	04	A0	1FFFFFFE	8F	CB	BICL3	#536870910, 4(R0), TEMP5	:	
5A	08	A0	1FFFFFFE	8F	CB	BICL3	#536870910, 8(R0), TEMP6	:	
59	0C	A0	1FFFFFFE	8F	CB	BICL3	#536870910, 12(R0), TEMP7	:	
			59	DD	007FD	PUSHL	TEMP7	:	
			5A	DD	007FF	PUSHL	TEMP6	:	
			5B	DD	00801	PUSHL	TEMP5	:	
	7E		55	3C	00803	MOVZWL	TEMP4, -(SP)	:	
		0000G	CF	9F	00806	PUSHAB	FMT_UBI_DUMP	:	
	00000000G	9F	05	FB	0080A	CALLS	#5, a#DS\$PRINTX	:	
		50	0000G	CF	D0	MOVL	UBE_BASE_ADDR, R0	:	2229
	0C	AE		60	B0	MOVW	(R0), TEMP0	:	
	08	AE	02	A0	B0	MOVW	2(R0), TEMP1	:	
	04	AE	04	A0	B0	MOVW	4(R0), TEMP2	:	
		6E	06	A0	B0	MOVW	6(R0), TEMP3	:	
		55	0E	A0	B0	MOVW	14(R0), TEMP4	:	
	7E		55	3C	0082C	MOVZWL	TEMP4, -(SP)	:	
	7E	04	AE	3C	0082F	MOVZWL	TEMP3, -(SP)	:	
	7E	0C	AE	3C	00833	MOVZWL	TEMP2, -(SP)	:	
	7E	14	AE	3C	00837	MOVZWL	TEMP1, -(SP)	:	
	7E	1C	AE	3C	0083B	MOVZWL	TEMP0, -(SP)	:	
	7E	0000G	CF	9A	0083F	MOVZBL	UBE_DRIVE, -(SP)	:	
		0000G	CF	9F	00844	PUSHAB	FMT_UBE_DUMP	:	
	00000000G	9F	07	FB	00848	CALLS	#7, a#DS\$PRINTX	:	
			21	11	0084F	BRB	65\$	:	2224
	1C	0000G	CF	E9	00851	BLBC	UBE_XFER_ERR, 65\$	:	2233
			7E	7C	00856	CLRQ	-(SP)	:	2236
			7E	7C	00858	CLRQ	-(SP)	:	
			7E	7C	0085A	CLRQ	-(SP)	:	
		0000G	CF	9F	0085C	PUSHAB	PRINT_DCOMP_ERR	:	
		0000G	CF	9F	00860	PUSHAB	UBIMOD_BDP	:	
	7E	0000G	CF	9A	00864	MOVZBL	LUN, -(SP)	:	
			0F	DD	00869	PUSHL	#15	:	

03	00000000G	9F	0A	FB	0086B	CALLS	#10, a#DS\$ERRHARD	:	2238
	0000G	CF	01	E0	00872	BBS	#1, UBE_STAT_ERR, 66\$	:	
			0094	31	00878	BRW	67\$	:	
			7E	7C	0087B	CLRQ	-(SP)	:	2241
			7E	7C	0087D	CLRQ	-(SP)	:	
			7E	7C	0087F	CLRQ	-(SP)	:	
			7E	D4	00881	CLRL	-(SP)	:	
			CF	9F	00883	PUSHAB	UBIMOD_BDP	:	
		7E	CF	9A	00887	MOVZBL	LUN, -(SP)	:	
			10	DD	0088C	PUSHL	#16	:	
	00000000G	9F	0A	FB	0088E	CALLS	#10, a#DS\$ERRHARD	:	
55		50	CF	D0	00895	MOVL	UBI_UNDER_TEST, R0	:	
5B	04	60	8F	AB	0089A	BICW3	#65534, (R0), TEMP4	:	
5A	08	A0	8F	CB	008A0	BICL3	#536870910, 4(R0), TEMP5	:	
59	0C	A0	8F	CB	008A9	BICL3	#536870910, 8(R0), TEMP6	:	
		A0	8F	CB	008B2	BICL3	#536870910, 12(R0), TEMP7	:	
			59	DD	008BB	PUSHL	TEMP7	:	
			5A	DD	008BD	PUSHL	TEMP6	:	
			5B	DD	008BF	PUSHL	TEMP5	:	
		7E	55	3C	008C1	MOVZWL	TEMP4, -(SP)	:	
			CF	9F	008C4	PUSHAB	FMT_UBI_DUMP	:	
	00000000G	9F	05	FB	008C8	CALLS	#5, a#DS\$PRINTX	:	2243
		50	CF	D0	008CF	MOVL	UBE_BASE_ADDR+4, R0	:	
	0C	AE	60	B0	008D4	MOVW	(R0), TEMP0	:	
	08	AE	02	A0	008D8	MOVW	2(R0), TEMP1	:	
	04	AE	04	A0	008DD	MOVW	4(R0), TEMP2	:	
		6E	06	A0	008E2	MOVW	6(R0), TEMP3	:	
		55	0E	A0	008E6	MOVW	14(R0), TEMP4	:	
		7E	55	3C	008EA	MOVZWL	TEMP4, -(SP)	:	
		7E	04	AE	008ED	MOVZWL	TEMP3, -(SP)	:	
		7E	0C	AE	008F1	MOVZWL	TEMP2, -(SP)	:	
		7E	14	AE	008F5	MOVZWL	TEMP1, -(SP)	:	
		7E	1C	AE	008F9	MOVZWL	TEMP0, -(SP)	:	
		7E	CF	9A	008FD	MOVZBL	UBE_DRIVE+1, -(SP)	:	
			CF	9F	00902	PUSHAB	FMT_UBE_DUMP	:	
	00000000G	9F	07	FB	00906	CALLS	#7, a#DS\$PRINTX	:	
			23	11	0090D	BRB	68\$	:	2238
1D	0000G	CF	01	E1	0090F	BBC	#1, UBE_XFER_ERR, 68\$	:	2247
			7E	7C	00915	CLRQ	-(SP)	:	2250
			7E	7C	00917	CLRQ	-(SP)	:	
		7E	01	7D	00919	MOVQ	#1, -(SP)	:	
			CF	9F	0091C	PUSHAB	PRINT_DCMP_ERR	:	
			CF	9F	00920	PUSHAB	UBIMOD_BDP	:	
		7E	CF	9A	00924	MOVZBL	LUN, -(SP)	:	
			11	DD	00929	PUSHL	#17	:	
	00000000G	9F	0A	FB	0092B	CALLS	#10, a#DS\$ERRHARD	:	
03	0000G	CF	02	E0	00932	BBS	#2, UBE_STAT_ERR, 69\$	:	2252
			0094	31	00938	BRW	70\$	:	
			7E	7C	0093B	CLRQ	-(SP)	:	2255
			7E	7C	0093D	CLRQ	-(SP)	:	
			7E	7C	0093F	CLRQ	-(SP)	:	
			7E	D4	00941	CLRL	-(SP)	:	
			CF	9F	00943	PUSHAB	UBIMOD_BDP	:	
		7E	CF	9A	00947	MOVZBL	LUN, -(SP)	:	
			12	DD	0094C	PUSHL	#18	:	

		00000000G	9F		0A	FB	0094E	CALLS	#10, a#DS\$ERRHARD	:	
			50	0000G	CF	D0	00955	MOVL	UBI_UNDER_TEST, R0	:	
55			60	FFFE	8F	AB	0095A	BICW3	#65534, (R0), TEMP4	:	
5B	04		A0	1FFFFFFE	8F	CB	00960	BICL3	#536870910, 4(R0), TEMP5	:	
5A	08		A0	1FFFFFFE	8F	CB	00969	BICL3	#536870910, 8(R0), TEMP6	:	
59	0C		A0	1FFFFFFE	8F	CB	00972	BICL3	#536870910, 12(R0), TEMP7	:	
					59	DD	0097B	PUSHL	TEMP7	:	
					5A	DD	0097D	PUSHL	TEMP6	:	
					5B	DD	0097F	PUSHL	TEMP5	:	
		7E			55	3C	00981	MOVZWL	TEMP4, -(SP)	:	
				0000G	CF	9F	00984	PUSHAB	FMT_UBI_DUMP	:	
		00000000G	9F		05	FB	00988	CALLS	#5, a#DS\$PRINTX	:	
			50	0000G	CF	D0	0098F	MOVL	UBE_BASE_ADDR+8, R0	:	2257
	0C		AE		60	B0	00994	MOVW	(R0), TEMP0	:	
	08		AE	02	A0	B0	00998	MOVW	2(R0), TEMP1	:	
	04		AE	04	A0	B0	0099D	MOVW	4(R0), TEMP2	:	
			6E	06	A0	B0	009A2	MOVW	6(R0), TEMP3	:	
			55	0E	A0	B0	009A6	MOVW	14(R0), TEMP4	:	
			7E		55	3C	009AA	MOVZWL	TEMP4, -(SP)	:	
			7E	04	AE	3C	009AD	MOVZWL	TEMP3, -(SP)	:	
			7E	0C	AE	3C	009B1	MOVZWL	TEMP2, -(SP)	:	
			7E	14	AE	3C	009B5	MOVZWL	TEMP1, -(SP)	:	
			7E	1C	AE	3C	009B9	MOVZWL	TEMP0, -(SP)	:	
			7E		CF	9A	009BD	MOVZBL	UBE_DRIVE+2, -(SP)	:	
				0000G	CF	9F	009C2	PUSHAB	FMT_UBE_DUMP	:	
		00000000G	9F		07	FB	009C6	CALLS	#7, a#DS\$PRINTX	:	
					23	11	009CD	BRB	71\$	:	2252
1D	0000G		CF		02	E1	009CF	BBC	#2, UBE_XFER_ERR, 71\$	:	2261
					7E	7C	009D5	CLRQ	-(SP)	:	2264
					7E	7C	009D7	CLRQ	-(SP)	:	
		7E			02	7D	009D9	MOVQ	#2, -(SP)	:	
				0000G	CF	9F	009DC	PUSHAB	PRINT_DCOMP_ERR	:	
				0000G	CF	9F	009E0	PUSHAB	UBIMOD_BDP	:	
			7E	0000G	CF	9A	009E4	MOVZBL	LUN, -(SP)	:	
					13	DD	009E9	PUSHL	#19	:	
FC39	52	00000000G	9F		0A	FB	009EB	CALLS	#10, a#DS\$ERRHARD	:	
			01		03	F1	009F2	ACBL	#3, #1, N, 52\$	:	2125
					7E	7C	009F8	CLRQ	-(SP)	:	2268
				0000G	CF	DD	009FA	PUSHL	UBE_PAGE_CNT	:	
					03	FB	009FE	CALLS	#3, a#DS\$RELBUF	:	
FBC8	20	AE	01		03	F1	00A05	ACBL	#3, #1, I, 45\$	:	2093
					0565	31	00A0C	BRW	106\$	:	1798
				20	AE	D4	00A0F	CLRL	I	:	2274
		0000G	CF	20	AE	90	00A12	MOVB	I, UBE_DP_NO	:	2276
			03	0000G	CF	91	00A18	CMPB	UBE_DP_NO, #3	:	2277
					06	12	00A1D	BNEQ	74\$	:	
				0000G	CF	94	00A1F	CLRB	UBE_DP_NO+1	:	2278
					08	11	00A23	BRB	75\$	:	
	0000G	CF	0000G	CF	01	81	00A25	ADDB3	#1, UBE_DP_NO, UBE_DP_NO+1	:	2279
			03	0000G	CF	91	00A2D	CMPB	UBE_DP_NO+1, #3	:	2281
					06	12	00A32	BNEQ	76\$	:	
				0000G	CF	94	00A34	CLRB	UBE_DP_NO+2	:	2282
					08	11	00A38	BRB	77\$	:	
	0000G	CF	0000G	CF	01	81	00A3A	ADDB3	#1, UBE_DP_NO+1, UBE_DP_NO+2	:	2283
			03	0000G	CF	91	00A42	CMPB	UBE_DP_NO+2, #3	:	2285

Address	Op-Code	Op-Code Hex	Op-Code Name	Comment
0000G	CF	0000G	CF	01000100
			51	0000GCF
			08	A1
F3			50	03
			06	12
			CF	94
			08	11
			01	81
			8F	D0
			50	D4
			40	D0
			A1	B4
			03	F3
				00A47
				00A49
				00A4D
				00A4F
				00A57
				00A60
				00A62
				00A68
				00A6B
				BNEQ
				78\$
				CLRB
				UBE_DP_NO+3
				BRB
				79\$
				ADDB3
				#1, UBE_DP_NO+2, UBE_DP_NO+3
				MOVL
				#16777472, UBE_MODE
				CLRL
				X
				MOVL
				UBE_BASE_ADDR[X], R1
				CLRW
				8(R1)
				AOBLEQ
				#3, X, 80\$

```
; This is the action taken for four UBEs. First, the map
; buffers routine is called. This routine allocates buffer spa
; ce
; for the transfers.
```

Address	Op Code	Op Name	Comments	PC
0000G	CF			
1C	AE			
01	DD 00A6F	PUSHL	#1	; 2307
04	DD 00A71	PUSHL	#4	;
02	FB 00A73	CALLS	#2, MAP_BUFFERS	;
50	D0 00A78	MOVL	R0, NO_MAPS_INUSE	;
03	12 00A7C	BNEQ	82\$	; 2308
50	D4 00A7E 81\$:	CLRL	R0	; 2309
	04 00A80	RET		;
52	D4 00A81 82\$:	CLRL	N	; 2312

```
; Next, set up the buffers.  The parameters are buffer number
; and the pattern to use.
```

Address	Op-Code	Op-Code Hex	Op-Code Dec	Op-Code Name	Comment	Address
0000G	7E	40	AE42	01 DD 00A83	PUSHL #1	2321
				3C 00A85	MOVZWL WORD_PATTERN_0[N], -(SP)	
			7E	D4 00A8A	- (SP)	
	CF		03	FB 00A8C	CALLS #3, BUFFER_SETUP	
			01	DD 00A91	PUSHL #1	2322
	7E	38	AE42	3C 00A93	MOVZWL WORD_PATTERN_1[N], -(SP)	
			01	DD 00A98	PUSHL #1	
	CF		03	FB 00A9A	CALLS #3, BUFFER_SETUP	
			01	DD 00A9F	PUSHL #1	2323
	7E	30	AE42	3C 00AA1	MOVZWL WORD_PATTERN_2[N], -(SP)	
			02	DD 00AA6	PUSHL #2	
	CF		03	FB 00AA8	CALLS #3, BUFFER_SETUP	
			01	DD 00AAD	PUSHL #1	2324
	7E	28	AE42	3C 00AAF	MOVZWL WORD_PATTERN_3[N], -(SP)	
			03	DD 00AB4	PUSHL #3	
	CF		03	FB 00AB6	CALLS #3, BUFFER_SETUP	
0000G	CF	3C	AE42	B0 00ABB	MOVW WORD_PATTERN_0[N], UBE_INIT_DATA	2325
0000G	CF	34	AE42	B0 00AC2	MOVW WORD_PATTERN_1[N], UBE_INIT_DATA+2	2326
0000G	CF	2C	AE42	B0 00AC9	MOVW WORD_PATTERN_2[N], UBE_INIT_DATA+4	2327
0000G	CF	24	AE42	B0 00AD0	MOVW WORD_PATTERN_3[N], UBE_INIT_DATA+6	2328

```
; Set up the UBEs and UBI for the transfer.  The parameters
; are UBE number, starting map number, offset mode, data path
; number, rotate data path mode, and data pattern if DATIP.
```

```

7E      0000G  7E  D4 00AD7      CLRL  -(SP)      ; 2338
7E      0000G  CF  9A 00AD9      MOVZBL UBE_DP_NO, -(SP) ;
7E      0000G  CF  3C 00ADE      MOVZWL UBE_MAP_NO, -(SP) ;

```



```

0000G CF 7E D4 00AE3 CLRL -(SP) ;
0000G CF 04 FB 00AE5 CALLS #4, XFER_SETUP ;
7E 0000G CF 7E D4 00AEA CLRL -(SP) ; 2339
7E 0000G CF 9A 00AEC MOVZBL UBE_DP_NO+1, -(SP) ;
0000G CF 01 DD 00AF6 MOVZWL UBE_MAP_NO+2, -(SP) ;
0000G CF 04 FB 00AF8 PUSHL #1 ;
7E 0000G CF 7E D4 00AFD CALLS #4, XFER_SETUP ;
7E 0000G CF 9A 00AFF CLRL -(SP) ; 2340
0000G CF 02 DD 00B09 MOVZBL UBE_DP_NO+2, -(SP) ;
7E 0000G CF 3C 00B04 MOVZWL UBE_MAP_NO+4, -(SP) ;
0000G CF 04 FB 00B0B PUSHL #2 ;
7E 0000G CF 7E D4 00B10 CALLS #4, XFER_SETUP ;
7E 0000G CF 9A 00B12 CLRL -(SP) ; 2341
0000G CF 03 DD 00B1C MOVZBL UBE_DP_NO+3, -(SP) ;
0000G CF 04 FB 00B1E MOVZWL UBE_MAP_NO+6, -(SP) ;
PUSHL #3 ;
CALLS #4, XFER_SETUP ;

; Set up the UBE data buffer and the expected data for the
; transfer analysis routine. Set the IPL to 16 to allow the
; UBE to interrupt on error. Then start the transfer.
;
00000000G 9F 00 FB 00B23 CALLS #0, @DS$BREAK ; 2350
00 0000G CF 50 D4 00B2A CLRL X ; 2352
00 0000G CF 50 E5 00B2C 84$: BBCC X, UBE_STAT_ERR, 85$ ; 2354
F0 0000G CF 50 E5 00B32 85$: BBCC X, UBE_XFER_ERR, 86$ ; 2355
50 03 F3 00B38 86$: AOBLEQ #3, X, 84$ ; 2352
56 0000G CF D0 00B3C MOVL UBE_BASE_ADDR, R6 ; 2357
18 AE 3C AE42 3C 00B41 MOVZWL WORD_PATTERN_0[N], 24(SP) ;
18 AE 18 AE D2 00B47 MCOML 24(SP), 24(SP) ;
66 18 AE B0 00B4C MOVW 24(SP), (R6) ;
54 0000G CF D0 00B50 MOVL UBE_BASE_ADDR+4, R4 ; 2358
14 AE 34 AE42 3C 00B55 MOVZWL WORD_PATTERN_1[N], 20(SP) ;
14 AE 14 AE D2 00B5B MCOML 20(SP), 20(SP) ;
64 14 AE B0 00B60 MOVW 20(SP), (R4) ;
51 0000G CF D0 00B64 MOVL UBE_BASE_ADDR+8, R1 ; 2359
10 AE 2C AE42 3C 00B69 MOVZWL WORD_PATTERN_2[N], 16(SP) ;
10 AE 10 AE D2 00B6F MCOML 16(SP), 16(SP) ;
61 10 AE B0 00B74 MOVW 16(SP), (R1) ;
50 0000G CF D0 00B78 MOVL UBE_BASE_ADDR+12, R0 ; 2360
58 24 AE42 3C 00B7D MOVZWL WORD_PATTERN_3[N], R8 ;
58 58 D2 00B82 MCOML R8, R8 ;
60 58 B0 00B85 MOVW R8, (R0) ;
0000G CF 18 AE B0 00B88 MOVW 24(SP), UBE_EXP_DATA ; 2361
0000G CF 14 AE B0 00B8E MOVW 20(SP), UBE_EXP_DATA+2 ; 2362
0000G CF 10 AE B0 00B94 MOVW 16(SP), UBE_EXP_DATA+4 ; 2363
0000G CF 58 B0 00B9A MOVW R8, UBE_EXP_DATA+6 ; 2364
06 A6 0663 8F B0 00B9F MOVW #1635, 6(R6) ; 2366
06 A4 0665 8F B0 00BA5 MOVW #1637, 6(R4) ; 2367
06 A1 0669 8F B0 00BAB MOVW #1641, 6(R1) ; 2368
06 A0 0671 8F B0 00BB1 MOVW #1649, 6(R0) ; 2369

; Wait for enough time to allow the transfer to complete.
; then check to see if a status error was indicated. If it
; was, then dump the UBI and UBE registers. If there a

```

; status error was not indicated, then check to see if there
; was a data error. If so, report it.

53	1C	AE	0000G	CF	D4	00BB7	CLRL	UBE_INT_OCC	; 2382
		12	0000G	CF	01	C3 00BBB	SUBL3	#1, NO MAPS_INUSE, CURRENT_MAP	; 2383
		0D	0000G	CF	E9	00BC0	BLBC	UBE_INT_OCC, 88\$	; 2386
		08	0000G	CF	E9	00BC5	BLBC	UBE_INT_OCC+1, 88\$	; 2387
		03	0000G	CF	E9	00BCA	BLBC	UBE_INT_OCC+2, 88\$	; 2387
			0000G	CF	E9	00BCF	BLBC	UBE_INT_OCC+3, 88\$	; 2387
			0086		31	00BD4	BRW	93\$	; 2387
	000001FF	8F			53	D1 00BD7	CMPL	CURRENT_MAP, #511	; 2389
					06	12 00BDE	BNEQ	89\$	; 2389
		53	1C	AE	D0	00BE0	MOVL	NO MAPS_INUSE, CURRENT_MAP	; 2390
					02	11 00BE4	BRB	90\$	; 2391
					53	D6 00BE6	INCL	CURRENT_MAP	; 2391
		50	0000G	D4	43	DE 00BE8	MOVAL	#UBI_UNDER_TEST[CURRENT_MAP], R0	; 2396
		56	0800	C0	9E	00BEE	MOVAB	2048(R0), R6	; 2396
					66	D4 00BF3	CLRL	(R6)	; 2397
57		66	7D9F8000	8F	CB	00BF5	BICL3	#2107604992, (R6), MAP_DATA	; 2397
					1F	13 00BFD	BEQL	91\$	; 2402
					7E	D4 00BFF	CLRL	-(SP)	; 2402
					57	DD 00C01	PUSHL	MAP_DATA	; 2402
					7E	D4 00C03	CLRL	-(SP)	; 2402
			0000G	CF	9F	00C05	PUSHAB	FMT_MAP_FAIL	; 2402
			0048	8F	BB	00C09	PUSHR	#M<R3,R6>	; 2402
			0000G	CF	9F	00C0D	PUSHAB	PRINT_MAP_ERR	; 2402
			0000G	CF	9F	00C11	PUSHAB	UBIMOD_BDP	; 2402
		7E	0000G	CF	9A	00C15	MOVZBL	LUN, -(SP)	; 2402
					14	DD 00C1A	PUSHL	#20	; 2402
					35	11 00C1C	BRB	92\$	; 2402
		66			01	CE 00C1E	MNEGL	#1, (R6)	; 2407
57		66	7D9F8000	8F	CB	00C21	BICL3	#2107604992, (R6), MAP_DATA	; 2408
82607FFF		8F			57	D1 00C29	CMPL	MAP_DATA, #-2107604993	; 2408
					8E	13 00C30	BEQL	87\$	; 2413
					7E	D4 00C32	CLRL	-(SP)	; 2413
					57	DD 00C34	PUSHL	MAP_DATA	; 2413
			82607FFF		8F	DD 00C36	PUSHL	#-2107604993	; 2413
			0000G	CF	9F	00C3C	PUSHAB	FMT_MAP_FAIL	; 2413
			0048	8F	BB	00C40	PUSHR	#M<R3,R6>	; 2413
			0000G	CF	9F	00C44	PUSHAB	PRINT_MAP_ERR	; 2413
			0000G	CF	9F	00C48	PUSHAB	UBIMOD_BDP	; 2413
		7E	0000G	CF	9A	00C4C	MOVZBL	LUN, -(SP)	; 2413
					15	DD 00C51	PUSHL	#21	; 2413
	00000000G	9F			0A	FB 00C53	CALLS	#10, a#DS\$ERRHARD	; 2384
					FF63	31 00C5A	BRW	87\$	; 2384
		03	0000G	CF	E8	00C5D	BLBS	UBE_STAT_ERR, 94\$	; 2418
			0094		31	00C62	BRW	95\$	; 2421
					7E	7C 00C65	CLRQ	-(SP)	; 2421
					7E	7C 00C67	CLRQ	-(SP)	; 2421
					7E	7C 00C69	CLRQ	-(SP)	; 2421
					7E	D4 00C6B	CLRL	-(SP)	; 2421
			0000G	CF	9F	00C6D	PUSHAB	UBIMOD_BDP	; 2421
		7E	0000G	CF	9A	00C71	MOVZBL	LUN, -(SP)	; 2421
					16	DD 00C76	PUSHL	#22	; 2421
	00000000G	9F			0A	FB 00C78	CALLS	#10, a#DS\$ERRHARD	; 2421

		50	0000G	CF	D0	00C7F	MOVL	UBI_UNDER_TEST, R0	:	
55		60	FFFE	8F	AB	00C84	BICW3	#65534, (R0), TEMP4	:	
5B	04	A0	1FFFFFFE	8F	CB	00C8A	BICL3	#536870910, 4(R0), TEMP5	:	
5A	08	A0	1FFFFFFE	8F	CB	00C93	BICL3	#536870910, 8(R0), TEMP6	:	
59	0C	A0	1FFFFFFE	8F	CB	00C9C	BICL3	#536870910, 12(R0), TEMP7	:	
				59	DD	00CA5	PUSHL	TEMP7	:	
				5A	DD	00CA7	PUSHL	TEMP6	:	
				5B	DD	00CA9	PUSHL	TEMP5	:	
		7E		55	3C	00CAB	MOVZWL	TEMP4, -(SP)	:	
			0000G	CF	9F	00CAE	PUSHAB	FMT_UBI_DUMP	:	
		9F		05	FB	00CB2	CALLS	#5, a#DS\$PRINTX	:	
		50	0000G	CF	D0	00CB9	MOVL	UBE_BASE_ADDR, R0	:	2423
	0C	AE		60	B0	00CBE	MOVW	(R0), TEMP0	:	
	08	AE	02	A0	B0	00CC2	MOVW	2(R0), TEMP1	:	
	04	AE	04	A0	B0	00CC7	MOVW	4(R0), TEMP2	:	
		6E	06	A0	B0	00CCC	MOVW	6(R0), TEMP3	:	
		55	0E	A0	B0	00CD0	MOVW	14(R0), TEMP4	:	
		7E		55	3C	00CD4	MOVZWL	TEMP4, -(SP)	:	
		7E	04	AE	3C	00CD7	MOVZWL	TEMP3, -(SP)	:	
		7E	0C	AE	3C	00CDB	MOVZWL	TEMP2, -(SP)	:	
		7E	14	AE	3C	00CDF	MOVZWL	TEMP1, -(SP)	:	
		7E	1C	AE	3C	00CE3	MOVZWL	TEMP0, -(SP)	:	
		7E	0000G	CF	9A	00CE7	MOVZBL	UBE_DRIVE, -(SP)	:	
			0000G	CF	9F	00CEC	PUSHAB	FMT_UBE_DUMP	:	
		9F		07	FB	00CF0	CALLS	#7, a#DS\$PRINTX	:	
				21	11	00CF7	BRB	96\$	:	2418
		1C	0000G	CF	E9	00CF9	BLBC	UBE_XFER_ERR, 96\$	:	2427
				7E	7C	00CFE	CLRQ	-(SP)	:	2430
				7E	7C	00D00	CLRQ	-(SP)	:	
				7E	7C	00D02	CLRQ	-(SP)	:	
			0000G	CF	9F	00D04	PUSHAB	PRINT_DCOMP_ERR	:	
			0000G	CF	9F	00D08	PUSHAB	UBIMOD_BDP	:	
		7E	0000G	CF	9A	00D0C	MOVZBL	LUN, -(SP)	:	
				17	DD	00D11	PUSHL	#23	:	
		9F		0A	FB	00D13	CALLS	#10, a#DS\$ERRHARD	:	
03		CF		01	E0	00D1A	BBS	#1, UBE_STAT_ERR, 97\$	:	2432
				0094	31	00D20	BRW	98\$	:	
				7E	7C	00D23	CLRQ	-(SP)	:	2435
				7E	7C	00D25	CLRQ	-(SP)	:	
				7E	7C	00D27	CLRQ	-(SP)	:	
				7E	D4	00D29	CLRL	-(SP)	:	
			0000G	CF	9F	00D2B	PUSHAB	UBIMOD_BDP	:	
		7E	0000G	CF	9A	00D2F	MOVZBL	LUN, -(SP)	:	
				18	DD	00D34	PUSHL	#24	:	
		9F		0A	FB	00D36	CALLS	#10, a#DS\$ERRHARD	:	
		50	0000G	CF	D0	00D3D	MOVL	UBI_UNDER_TEST, R0	:	
55		60	FFFE	8F	AB	00D42	BICW3	#65534, (R0), TEMP4	:	
5B	04	A0	1FFFFFFE	8F	CB	00D48	BICL3	#536870910, 4(R0), TEMP5	:	
5A	08	A0	1FFFFFFE	8F	CB	00D51	BICL3	#536870910, 8(R0), TEMP6	:	
59	0C	A0	1FFFFFFE	8F	CB	00D5A	BICL3	#536870910, 12(R0), TEMP7	:	
				59	DD	00D63	PUSHL	TEMP7	:	
				5A	DD	00D65	PUSHL	TEMP6	:	
				5B	DD	00D67	PUSHL	TEMP5	:	
		7E		55	3C	00D69	MOVZWL	TEMP4, -(SP)	:	
			0000G	CF	9F	00D6C	PUSHAB	FMT_UBI_DUMP	:	

00000000G	9F		05	FB	00D70	CALLS	#5, a#DS\$PRINTX	:	
	50	0000G	CF	D0	00D77	MOVL	UBE_BASE_ADDR+4, R0	:	2437
0C	AE		60	B0	00D7C	MOVW	(R0), TEMP0	:	
08	AE	02	A0	B0	00D80	MOVW	2(R0), TEMP1	:	
04	AE	04	A0	B0	00D85	MOVW	4(R0), TEMP2	:	
	6E	06	A0	B0	00D8A	MOVW	6(R0), TEMP3	:	
	55	0E	A0	B0	00D8E	MOVW	14(R0), TEMP4	:	
	7E		55	3C	00D92	MOVZWL	TEMP4, -(SP)	:	
	7E	04	AE	3C	00D95	MOVZWL	TEMP3, -(SP)	:	
	7E	0C	AE	3C	00D99	MOVZWL	TEMP2, -(SP)	:	
	7E	14	AE	3C	00D9D	MOVZWL	TEMP1, -(SP)	:	
	7E	1C	AE	3C	00DA1	MOVZWL	TEMP0, -(SP)	:	
	7E	0000G	CF	9A	00DA5	MOVZBL	UBE_DRIVE+1, -(SP)	:	
		0000G	CF	9F	00DAA	PUSHAB	FMT_UBE_DUMP	:	
00000000G	9F		07	FB	00DAE	CALLS	#7, a#DS\$PRINTX	:	
			23	11	00DB5	BRB	99\$	:	2432
1D	0000G	CF	01	E1	00DB7	BBC	#1, UBE_XFER_ERR, 99\$	:	2441
			7E	7C	00DBD	CLRQ	-(SP)	:	2444
			7E	7C	00DBF	CLRQ	-(SP)	:	
	7E		01	7D	00DC1	MOVQ	#1, -(SP)	:	
		0000G	CF	9F	00DC4	PUSHAB	PRINT_DCOMP_ERR	:	
		0000G	CF	9F	00DC8	PUSHAB	UBIMOD_BDP	:	
	7E	0000G	CF	9A	00DCC	MOVZBL	LUN, -(SP)	:	
			19	DD	00DD1	PUSHL	#25	:	
00000000G	9F		0A	FB	00DD3	CALLS	#10, a#DS\$ERRHARD	:	
03	0000G	CF	02	E0	00DDA	BBS	#2, UBE_STAT_ERR, 100\$	:	2446
			0094	31	00DE0	BRW	101\$	:	
			7E	7C	00DE3	CLRQ	-(SP)	:	2449
			7E	7C	00DE5	CLRQ	-(SP)	:	
			7E	7C	00DE7	CLRQ	-(SP)	:	
			7E	D4	00DE9	CLRL	-(SP)	:	
		0000G	CF	9F	00DEB	PUSHAB	UBIMOD_BDP	:	
	7E	0000G	CF	9A	00DEF	MOVZBL	LUN, -(SP)	:	
			1A	DD	00DF4	PUSHL	#26	:	
00000000G	9F		0A	FB	00DF6	CALLS	#10, a#DS\$ERRHARD	:	
	50	0000G	CF	D0	00DFD	MOVL	UBI_UNDER_TEST, R0	:	
55	60	FFFE	8F	AB	00E02	BICW3	#65534, (R0), TEMP4	:	
5B	04	A0	1FFFFFFE	8F	CB	00E08	BICL3	#536870910, 4(R0), TEMP5	:
5A	08	A0	1FFFFFFE	8F	CB	00E11	BICL3	#536870910, 8(R0), TEMP6	:
59	0C	A0	1FFFFFFE	8F	CB	00E1A	BICL3	#536870910, 12(R0), TEMP7	:
			59	DD	00E23	PUSHL	TEMP7	:	
			5A	DD	00E25	PUSHL	TEMP6	:	
			5B	DD	00E27	PUSHL	TEMP5	:	
	7E		55	3C	00E29	MOVZWL	TEMP4, -(SP)	:	
		0000G	CF	9F	00E2C	PUSHAB	FMT_UBI_DUMP	:	
00000000G	9F		05	FB	00E30	CALLS	#5, a#DS\$PRINTX	:	
	50	0000G	CF	D0	00E37	MOVL	UBE_BASE_ADDR+8, R0	:	2451
	0C		60	B0	00E3C	MOVW	(R0), TEMP0	:	
	08	02	A0	B0	00E40	MOVW	2(R0), TEMP1	:	
	04	04	A0	B0	00E45	MOVW	4(R0), TEMP2	:	
	6E	06	A0	B0	00E4A	MOVW	6(R0), TEMP3	:	
	55	0E	A0	B0	00E4E	MOVW	14(R0), TEMP4	:	
	7E		55	3C	00E52	MOVZWL	TEMP4, -(SP)	:	
	7E	04	AE	3C	00E55	MOVZWL	TEMP3, -(SP)	:	
	7E	0C	AE	3C	00E59	MOVZWL	TEMP2, -(SP)	:	

	7E	14	AE	3C	00E5D	MOVZWL	TEMP1, -(SP)	:		
	7E	1C	AE	3C	00E61	MOVZWL	TEMP0, -(SP)	:		
	7E	0000G	CF	9A	00E65	MOVZBL	UBE_DRIVE+2, -(SP)	:		
		0000G	CF	9F	00E6A	PUSHAB	FMT_UBE_DUMP	:		
	00000000G	9F	07	FB	00E6E	CALLS	#7, @DS\$PRINTX	:		
			23	11	00E75	BRB	102\$	:	2446	
1D	0000G	CF	02	E1	00E77	BBC	#2, UBE_XFER_ERR, 102\$	:	2455	
			7E	7C	00E7D	CLRQ	-(SP)	:	2458	
			7E	7C	00E7F	CLRQ	-(SP)	:		
	7E		02	7D	00E81	MOVQ	#2, -(SP)	:		
		0000G	CF	9F	00E84	PUSHAB	PRINT_DCMF_ERR	:		
		0000G	CF	9F	00E88	PUSHAB	UBIMOD_BDP	:		
	7E	0000G	CF	9A	00E8C	MOVZBL	LUN, -(SP)	:		
			1B	DD	00E91	PUSHL	#27	:		
	00000000G	9F	0A	FB	00E93	CALLS	#10, @DS\$ERRHARD	:		
03	0000G	CF	03	E0	00E9A	BBS	#3, UBE_STAT_ERR, 103\$	:	2460	
			0094	31	00EA0	BRW	104\$	:		
			7E	7C	00EA3	CLRQ	-(SP)	:	2463	
			7E	7C	00EA5	CLRQ	-(SP)	:		
			7E	7C	00EA7	CLRQ	-(SP)	:		
			7E	D4	00EA9	CLRL	-(SP)	:		
		0000G	CF	9F	00EAB	PUSHAB	UBIMOD_BDP	:		
	7E	0000G	CF	9A	00EAF	MOVZBL	LUN, -(SP)	:		
			1C	DD	00EB4	PUSHL	#28	:		
	00000000G	9F	0A	FB	00EB6	CALLS	#10, @DS\$ERRHARD	:		
		50	0000G	CF	D0	00EBD	MOVL	UBI_UNDER_TEST, R0	:	
55		60	FFFE	8F	AB	00EC2	BICW3	#65534, (R0), TEMP4	:	
5B	04	A0	1FFFFFFE	8F	CB	00EC8	BICL3	#536870910, 4(R0), TEMP5	:	
5A	08	A0	1FFFFFFE	8F	CB	00ED1	BICL3	#536870910, 8(R0), TEMP6	:	
59	0C	A0	1FFFFFFE	8F	CB	00EDA	BICL3	#536870910, 12(R0), TEMP7	:	
			59	DD	00EE3	PUSHL	TEMP7	:		
			5A	DD	00EE5	PUSHL	TEMP6	:		
			5B	DD	00EE7	PUSHL	TEMP5	:		
	7E		55	3C	00EE9	MOVZWL	TEMP4, -(SP)	:		
		0000G	CF	9F	00EEC	PUSHAB	FMT_UBI_DUMP	:		
	00000000G	9F	05	FB	00EF0	CALLS	#5, @DS\$PRINTX	:		
		50	0000G	CF	D0	00EF7	MOVL	UBE_BASE_ADDR+12, R0	:	2465
	0C	AE	60	B0	00EFC	MOVW	(R0), TEMP0	:		
	08	AE	02	A0	B0	00F00	MOVW	2(R0), TEMP1	:	
	04	AE	04	A0	B0	00F05	MOVW	4(R0), TEMP2	:	
		6E	06	A0	B0	00F0A	MOVW	6(R0), TEMP3	:	
	55	0E	A0	B0	00F0E	MOVW	14(R0), TEMP4	:		
	7E		55	3C	00F12	MOVZWL	TEMP4, -(SP)	:		
	7E	04	AE	3C	00F15	MOVZWL	TEMP3, -(SP)	:		
	7E	0C	AE	3C	00F19	MOVZWL	TEMP2, -(SP)	:		
	7E	14	AE	3C	00F1D	MOVZWL	TEMP1, -(SP)	:		
	7E	1C	AE	3C	00F21	MOVZWL	TEMP0, -(SP)	:		
	7E	0000G	CF	9A	00F25	MOVZBL	UBE_DRIVE+3, -(SP)	:		
		0000G	CF	9F	00F2A	PUSHAB	FMT_UBE_DUMP	:		
	00000000G	9F	07	FB	00F2E	CALLS	#7, @DS\$PRINTX	:		
			23	11	00F35	BRB	105\$	:	2460	
1D	0000G	CF	03	E1	00F37	BBC	#3, UBE_XFER_ERR, 105\$	:	2469	
			7E	7C	00F3D	CLRQ	-(SP)	:	2472	
			7E	7C	00F3F	CLRQ	-(SP)	:		
	7E		03	7D	00F41	MOVQ	#3, -(SP)	:		

ZZ-ECCBA-1.4
UBI_TESTS_27_32
01-04

TEST_032: UBE Block Transfer Test
TEST_032: UBE Block Transfer Test

C 10

6-Sep-1985

Fiche 3 Frame C10

Sequence 531

6-Sep-1985 17:25:18

VAX-11 Bliss-32 V4.1-746

Page 100

6-Sep-1985 17:15:07

[LEMIEUX.ECCBA.RELEASE]ECCBA8.B32;19

(8)

			0000G	CF	9F	00F44	PUSHAB	PRINT_DCOMP_ERR	:		
			0000G	CF	9F	00F48	PUSHAB	UBIMOD_BDP	:		
		7E	0000G	CF	9A	00F4C	MOVZBL	LUN, -(SP)	:		
				1D	DD	00F51	PUSHL	#29	:		
FB23	52	00000000G	9F	0A	FB	00F53	CALLS	#10, a#DS\$ERRHARD	:		
			01	03	F1	00F5A	105\$:	ACBL	#3, #1, N, 83\$	: 2312	
				7E	7C	00F60	CLRQ	-(SP)	: 2476	:	
			0000G	CF	DD	00F62	PUSHL	UBE_PAGE_CNT	:		
				03	FB	00F66	CALLS	#3, a#DS\$RELBUF	:		
FA9E	20	AE	00000000G	9F	03	F1	00F6D	ACBL	#3, #1, I, 73\$	: 2274	
			00000000G	9F	00	FB	00F74	106\$:	CALLS	#0, a#DS\$BREAK	: 2484
				50	01	D0	00F7B	MOVL	#1, R0	: 2486	
					04	00F7E	RET		:	:	

; Routine Size: 3967 bytes, Routine Base: TEST_032 + 0000

ZZ-ECCBA-1.4
UBI_TESTS_27_32
01-04

TEST_032: UBE Block Transfer Test
TEST_032: UBE Block Transfer Test

D 10

6-Sep-1985

Fiche 3

Frame D10

Sequence 532

6-Sep-1985 17:25:18

VAX-11 Bliss-32 V4.1-746

Page 101

6-Sep-1985 17:15:07

[LEMIEUX.ECCBA.RELEASE]ECCBA8.B32;19

(9)

; 2487 1 \$DS_ENDMOD;
; 2488 1 END !End of module
; 2489 0 ELUDOM

.PSECT \$TSTCNT,NOWRT,NOEXE, OVR,2

00000020 00000 L_TSTCNT:

.LONG 32

PSECT SUMMARY

Name	Bytes	Attributes
\$OWN\$	22	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	176	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
DISPATCH	144	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_027	1466	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_028	1612	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_029	989	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_030	121	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_031	140	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
TEST_032	3967	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$TSTCNT	4	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, OVR, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
SYS\$COMMON:[SYSLIB]STARLET.L32;3	10093	1	0	598	00:00.8
SYS\$COMMON:[SYSLIB]DIAG.L32;265	784	46	5	85	00:00.3

COMMAND QUALIFIERS

; BLISS/LIST ECCBA8.B32

; Size: 8295 code + 346 data bytes
; Run Time: 01:25.2
; Elapsed Time: 01:39.8
; Lines/CPU Min: 1752
; Lexemes/CPU-Min: 29615

ZZ-ECCBA-1.4 TEST_032: UBE Block Transfer Test
UBI TESTS_27_32
01-04 TEST_032: UBE Block Transfer Test

; Memory Used: 1011 pages
; Compilation Complete

E 10

6-Sep-1985

Fiche 3

Frame E10

Sequence 533

6-Sep-1985 17:25:18

VAX-11 Bliss-32 V4.1-746

Page 102